

T
005 86
CHA
P.2



ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Proyecto de Graduación

**“Análisis, Diseño e Implementación de
un Sistema de alquiler de autos
usando tecnología Cliente Servidor
con arquitectura CORBA”**

**Previo a la obtención del Título de
*INGENIERO EN COMPUTACION***

Presentado por

***CAROLINA ELIZABETH CHANG HERRERA
BORIS HERNÁN MONTIEL RIVERA
LUIS ANGEL MUÑOZ CALLE***

**GUAYAQUIL – ECUADOR
1999**

AGRADECIMIENTO

A Dios por habernos ayudado en los momentos difíciles de nuestra carrera.

Al Ing. Carlos Valero, por su valiosa colaboración en el desarrollo del presente trabajo.

A todas aquellas personas que desinteresadamente contribuyeron en diversa forma a la realización práctica.

DEDICATORIA

A DIOS

A nuestros padres

A nuestros hermanos

A nuestros familiares

DECLARACION EXPRESA

“La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, nos corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL”

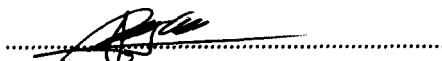
(Reglamento de Exámenes y Títulos profesionales de la ESPOL)



Carolina Elizabeth Chang Herrera



Boris Hernán Montiel Rivera

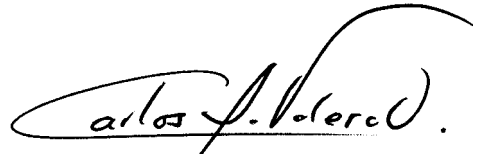


Luis Angel Muñoz Calle

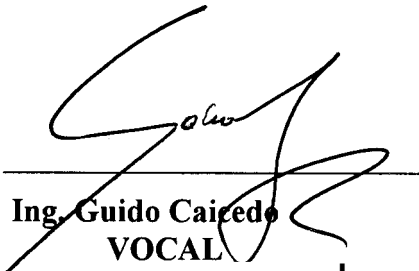
TRIBUNAL DE GRADUACION



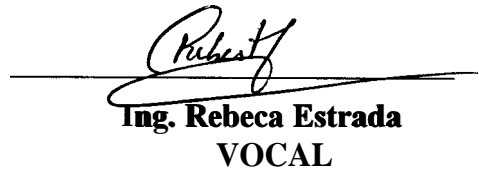
Ing. Carlos Monsalve
SUB-DECANO DE LA FIEC



Ing. Carlos Valero
DIRECTOR DEL TOPICO



Ing. Guido Caicedo
VOCAL



Ing. Rebeca Estrada
VOCAL

RESUMEN

A medida que el tiempo ha ido avanzando, las necesidades tecnológicas y de desarrollo comercial se han ido incrementando, tanto para las grandes empresas como también para pequeños industriales.

CORBA (Common Object Request Broker Architecture) ha suplido gran parte de esas necesidades, siendo una tecnología que permite el desarrollo de ambientes distribuidos, con gran despliegue y efectividad en situaciones donde las herramientas tradicionales no son lo suficientemente confiables y versátiles.

Para demostrar la aplicabilidad de CORBA, en el presente trabajo se ha elaborado un Sistema de Reservación de vehículos de una oficina de Renta de autos.

El análisis y diseño de nuestro Proyecto, obedece a la tecnología orientada a objetos (Diseño orientado a objetos), y se han implementado las siguientes transacciones: Reservación de un vehículo, Cancelación de reservación de vehículo, Entrega del vehículo y Devolución del vehículo, además de transacciones administrativas como son: Ingreso de Nuevos vehículos y Dar de baja a vehículos dañados, Seguridad de la Administración.

INDICE GENERAL

<i>RESUMEN</i>	<i>VI</i>
<i>INDICE GENERAL</i>	<i>VII</i>
<i>INDICE DE FIGURAS</i>	<i>X</i>
<i>INTRODUCCION</i>	<i>1</i>
<i>Capítulo 1</i>	<i>3</i>
MARCO TEORICO	3
1.1 Definición de la tecnología Cliente/Servidor	3
1.2 Esquema Cliente/Servidor Básico	4
1.2.1 Características deseables del esquema Cliente/Servidor	5
1.2.2 Limitaciones del esquema Cliente/Servidor básico	6
1.3 Esquema Cliente/Servidor de Procesamiento Distribuido	6
1.4 Middleware	8
1.5 Tecnología CORBA	9
1.5.1 IIOP (Internet Inter-ORB Protocol)	13
1.5.2 ORB (Object Request Broker)	15
1.5.2.1 Localizando el ORB	18
1.5.2.2 Inicialización de un objeto	19

1.5.2.3 Activación de los objetos	22
-----------------------------------	----

Capítulo 2 **25**

DESCRIPCION DEL PROYECTO **25**

2.1 Objetivos	25
---------------	----

2.2 Justificación	25
-------------------	----

2.3 Especificaciones del proyecto	26
-----------------------------------	----

2.4 Descripción general del proyecto	28
--------------------------------------	----

2.4.1 Detalles del Cliente	29
----------------------------	----

2.4.2 Detalles del Servidor	30
-----------------------------	----

2.4.3 Procesos involucrados en el Sistema	32
---	----

2.4.4 Descripción de la Persistencia del proyecto	34
---	----

Capítulo 3 **36**

ANALISIS Y DISEÑO DEL SISTEMA **36**

3.1 Introducción	36
------------------	----

3.2 Análisis de requerimientos	37
--------------------------------	----

3.3 Casos de Uso	38
------------------	----

3.3.1 Lista de casos de Uso	38
-----------------------------	----

3.3.2 Documentación de casos de uso	39
-------------------------------------	----

3.4 Lista de Objetos	43
----------------------	----

3.4 Definición de Escenarios	46
------------------------------	----

3.4.1 Alquiler	46
----------------	----

3.4.2 Administración	51
3.5 Diagrama de Interacción de Objetos (DIOs)	56
3.6 Modelo de Clases	57
3.7 Flujo de ventanas	58
3.8 Layouts de ventanas	59
3.9 Plan de pruebas	63
<i>Conclusiones</i>	69
<i>Glosario de Términos</i>	70
<i>Bibliografía</i>	72

INDICE DE FIGURAS

Figura 1. Esquema basico Cliente/Servidor	4
Figura 2. Componentes de un Sistema Cliente/Servidor	8
Figura 3. Independencia del lenguaje	10
Figura 4. Componentes de OMA	11
Figura 5. Comunicacion a traves de IIOP	14
Figura 6. Estructura de CORBA 2.0 ORB	16
Figura 7. Escenario de inicialización	20
Figura 8. Esquema de 3 capas con CORBA	30
Figura 9. Procesos involucrados en el Sistema	32

INTRODUCCION

Antecedentes

La tecnología Cliente/Servidor, ha evolucionado como una respuesta a la creciente necesidad de desarrollar Sistemas de información completamente distribuidos. En el camino, hemos visto como esta tecnología ha ido del esquema llamado “Two-Tier”, que abarca dos capas o niveles, hasta llegar a esquemas mas complejos llamados “Multi-Tier” o “N-Tier” que implica multiples capas o niveles.

CORBA es una arquitectura que justamente emplea el esquema “N-Tier”, teniendo actualmente una gran acogida para el desarrollo de aplicaciones distribuidas, respondiendo a las necesidades de interoperabilidad y permitiendo que las aplicaciones se comuniquen unas con otras sin importar donde se encuentren localizadas físicamente. En la actualidad se encuentra redefiniendo la forma en que desarrollamos, implementamos y mantenemos nuestras aplicaciones Cliente/Servidor.

CORBA emplea la metodología Orientada a objetos, teniendo todas sus ventajas en cuanto a flexibilidad y despliegue obtenidos con la misma.

Metodología para el desarrollo del Proyecto

Para el desarrollo de nuestro Proyecto, hemos seguido la siguiente metodología:

Inicialmente recuperamos la información requerida para la funcionalidad del sistema, por medio de entrevistas y reuniones con el personal de una oficina de Alquiler de vehículos, a fin de recoger sus requerimientos y necesidades del sistema.

Luego, realizamos el análisis y diseño orientado a objetos, utilizando la técnica OMT (Rumbaugh) y Jacobson.

Finalmente, implementamos el proyecto, utilizando las herramientas VisualAge para Java, Lotus Domino Web Server, Web Sphere Application Server, Base de datos DB2, Windows NT y 95, y VisiBroker para Java.

Capítulo 1

MARCO TEORICO

1.1 Definición de la tecnología Cliente/Servidor

Cliente/Servidor es una arquitectura de desarrollo de software que divide a la aplicación en al menos dos componentes: el proceso servidor, que puede ejecutarse en variadas plataformas computacionales y los procesos clientes, los cuales se comunican sobre redes de Área local o Área extendida, utilizando uno o varios protocolos de comunicación.

Con esta tecnología, se puede lograr, que varias computadoras compartan información de una forma transparente, aun cuando cada computador tenga un sistema operativo diferente.

1.2 Esquema Cliente/Servidor Basico

La arquitectura Cliente/Servidor ha evolucionado desde los esquemas mas sencillos (Figura 1) hasta los mas complejos, teniendo en la actualidad al Web como su maxima expresion. Este esquema es conocido como 2-Tier (2 capas).

En este esquema los clientes (desktops) accesan a los recursos y Aplicaciones ubicados en el servidor y la base de datos es accesada desde los desktops empleando componentes en ambas partes: en el servidor y en el cliente.

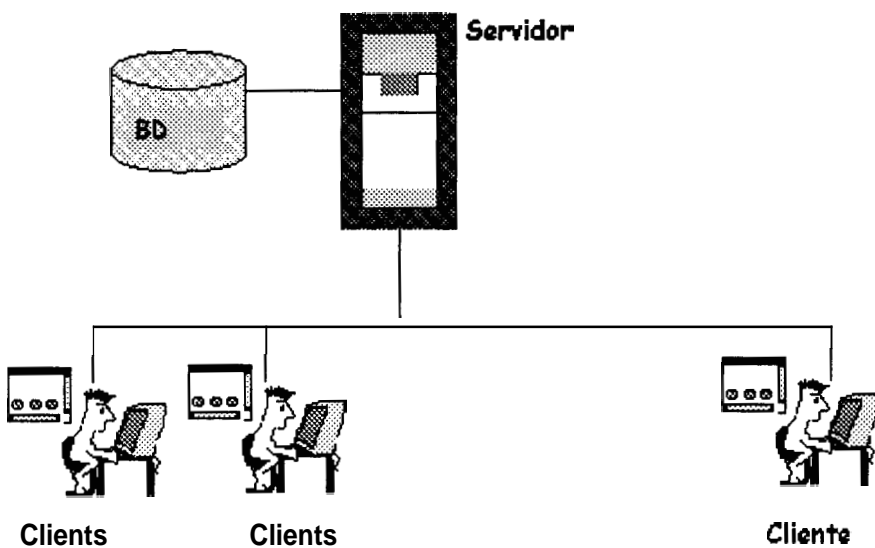


Figura 1. Esquema Basico Cliente/Servidor

El desarrollo de aplicaciones Cliente/Servidor requiere procesamiento de transacciones, diseño de base de datos, experiencia en comunicaciones y la interface grafica para el usuario. Aplicaciones mas avanzadas necesitan de conocimientos de objetos distribuidos y de Internet.

1.2.1 Caracteristicas deseables del esquema Cliente/Servidor

- 1. Transparencia de localización.-** El servidor es un proceso que puede residir en la misma maquina que el cliente ó en diferentes maquinas a traves de la red. Cliente/Servidor usualmente enmascara la localización del servidor, redireccionando las llamadas de servicio cuando se necesite.
- 2. Transparencia de Plataforma.-** El software Cliente/Servidor debe ser independiente del hardware o del Sistema operativo.
- 3. Escalabilidad.-** Los sistemas pueden ser escalados vertical u horizontalmente. Escalar horizontalmente significa añadir o remover clientes con solo un pequieio impacto en el performance. Escalar verticalmente significa migrar a un servidor mas rapido y grande o a multiservidores.

1.2.2 Limitaciones del esquema Cliente/Servidor basico

El esquema Cliente/Servidor basico tiene ciertas limitaciones como:

- ◆ Los clientes deben conocer la direccion del servidor al cual se pide el requerimiento.
- ◆ Para conocer dicha direccion se envía un mensaje broadcast, lo cual genera trafico en la red que congestiona la interface de red.
- ◆ Como solo existe un servidor atendiendo a todos los clientes, se crea un “cuello de botella”, con lo que aumenta el tiempo de respuesta a los requerimientos, y se sobrecarga la tarjeta de red, ya que se sobrecarga al servidor.

1.3 Esquema Cliente/Servidor de Procesamiento Distribuido

La evolución de la informática exige la ruptura de las fronteras de la comunicacion y la localidad física de las maquinas, gracias a lo cual, se ha logrado el procesamiento distribuido de las aplicaciones Cliente/Servidor. Este esquema es el resultado de la combinación de:

Modelo Cliente/Servidor puro.- Los clientes accesan a los recursos y aplicaciones del servidor.

Modelo Peer-Processing.- Cualquier nodo en la red puede ser cliente o servidor, incluso a veces simultaneamente, siendo cliente de un servicio y servidor de otro. El desarrollo se puede hacer con programacion pura y usando los protocolos de comunicacion para enviar informacion.

Modelo 3-Tier.- Este ultimo esquema es el que se encarga de monitorear la ejecución de procesos servidores y balancear su carga de trabajo. Los programadores no tienen que preocuparse de fallas, concurrencia, balanceo ante la carga y sincronizacion de recursos a traves de multiples nodos. Todo esto es transparente.

Las aplicaciones Cliente/Servidor “Multi-tier”, requieren de un gran conocimiento de objetos distribuidos, para maximizar su utilidad y disminuir la complejidad de las mismas, como la ubicacion fisica de los datos y los procesos servidores no es importante, se requiere la implementación completa del Directorio de servicios, el mismo que aísla la aplicacion de conocer donde se encuentran los procesos y los datos.

1.4 Middleware

Middleware es todo proceso intermedio entre el cliente y el servidor, que permite la interconexión y la comunicación entre ambos. Realiza tareas y servicios específicos. Apoya a los Sistemas operativos con funciones y procesos que extienden la capacidad y el alcance de los mismos. No incluye la lógica del proceso, lo cual es función principal del servidor; ni tampoco incluye interfaces del usuario que se encuentran en el cliente.

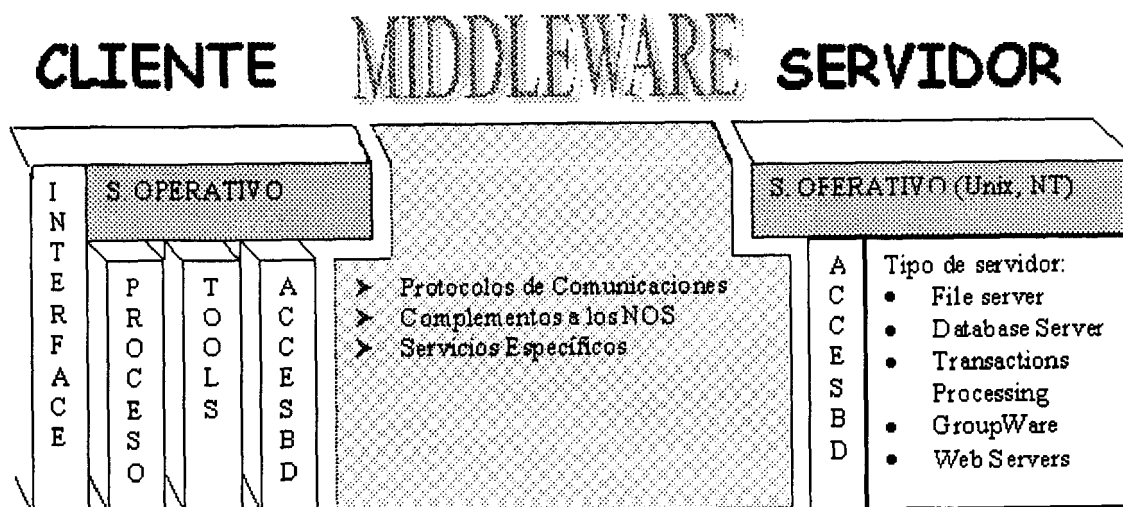


Figura 2. Componentes de un sistema Cliente/Servidor

Existen diversas formas de Middleware. Entre ellas tenemos las siguientes:

Mecanismos de comunicación.- Son los protocolos al nivel de sesión, transporte y red, que permiten establecer sesiones entre clientes y servidores, por ejemplo: TCP/IP (Sockets), NetBEUI/NetBIOS.

Funciones especiales de los NOS.- Network Operating System (NOS), son funciones y procesos de apoyo que extienden la capacidad de los sistemas operativos, con: funciones de seguridad, funciones de administración de archivos distribuidos, funciones para la invocación remota de procedimientos y procesos servidores.

De Servicios Específicos.- Incluye todas las otras funciones utilizadas, tales como funciones ligadas y dependientes principalmente del tipo de servidor.

1.5 Tecnología CORBA

CORBA, Common Object Request Broker Architecture, fue introducido en 1989 por el grupo OMG (Object Management Group), el mismo que no tiene fines de lucro y fue

creado con el propósito de promover el uso de la tecnología orientada a objetos en sistemas distribuidos, esto lo realiza a través de estándares que permiten la interoperabilidad y la portabilidad de aplicaciones orientadas a objetos.

El diseño CORBA se basa en el modelo de objetos de OMG. En este modelo los clientes requieren servicios de los objetos servidores a través de la interface IDL (Interface Definition Language) y el API (Application Programming Interfaces) que permiten la interacción Cliente/Servidor dentro de la implementación específica del ORB (Object Request Broker), sin importar el lenguaje de programación en el que este escrito el cliente o el servidor (Figura 3)

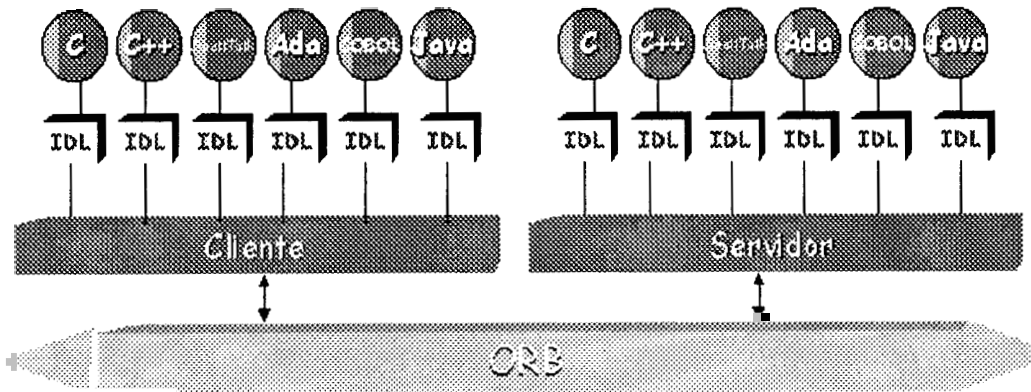


Figura 3. Independencia del Lenguaje

La OMG no produce lineamientos de software o implementación, solo especificaciones para aplicaciones. Esto lo hacen a través de OMA (Object Management Architecture), que tiene como su principal especificación a CORBA (Common Object Request Broker Architecture). OMA consiste de 4 componentes (Figura 4), que pueden ser divididos en dos grupos: el primer grupo son los componentes orientados al sistema: Object Request Broker y Object Services. El segundo grupo comprende los componentes orientados a la aplicación: Application objects y Common Facilities.

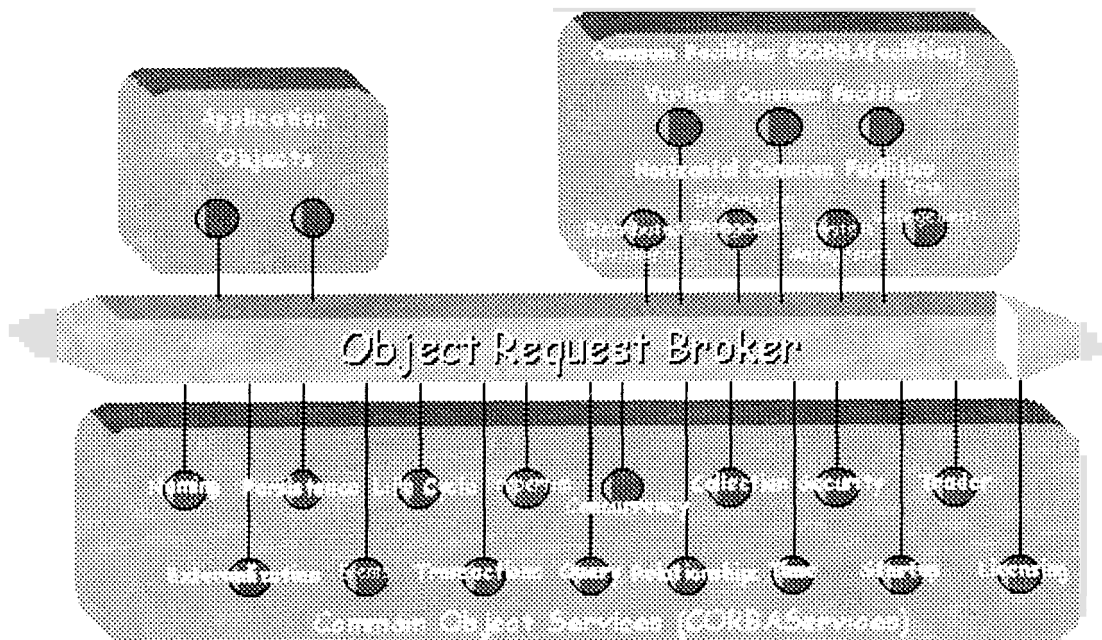


Figura 4. Componentes de OMA (Object Management Architecture)

De todos los componentes mencionados, el **Object Request Broker** es el que se encarga de la comunicación entre los diferentes componentes, permite a los objetos interactuar en un ambiente distribuido, heterogeneo, independiente de las plataformas en las cuales los objetos existen y de las técnicas usadas para su implementación.

El componente **Object Services** se encarga de la administración general de los objetos, esto es, crear objetos, control de acceso, mantener la pista de un objeto, etc. Ejemplos de ellos son:

- ◆ Naming Service.- que permite encontrar a los objetos por su nombre.
- ◆ Trading Service.- que permite a los clientes encontrar objetos, basado en sus propiedades.

Los componentes **Common Facilities** y **Application Objects** son los que en sus funciones invocan servicios de los componentes del sistema.

Los objetos CORBA se diferencian de los objetos típicos en las siguientes formas:

- ◆ Pueden estar presentes en cualquier parte de la red.
- ◆ Pueden interoperar con objetos de otras plataformas.

- ◆ Pueden escribirse en cualquier lenguaje de programación, ya que existe el mapeo de lenguajes a través del IDL.

CORBA especifica una solución que brinda interoperabilidad entre objetos en un ambiente heterogéneo y distribuido, de forma transparente para el programador.

1.5.1 IIOP (Internet Inter-ORB Protocol)

OMG ha definido un conjunto de reglas que dan formato a los datos que se transfieren, llamado CDR (Common Data Representation), además de un conjunto de tipos de mensajes. Juntos, los CDR y los tipos de mensajes, constituyen un protocolo abstracto llamado GIOP (General Inter-ORB Protocol). Para lograr verdadera interoperabilidad entre objetos distribuidos que residen en ambientes heterogéneos sobre Internet, se necesita que ORBs envíen mensajes a través de TCP/IP, porque es el protocolo de transporte orientado a conexión estándar para Internet. En pocas palabras, al unir GIOP + TCP/IP da como resultado el protocolo IIOP (Figura 5).

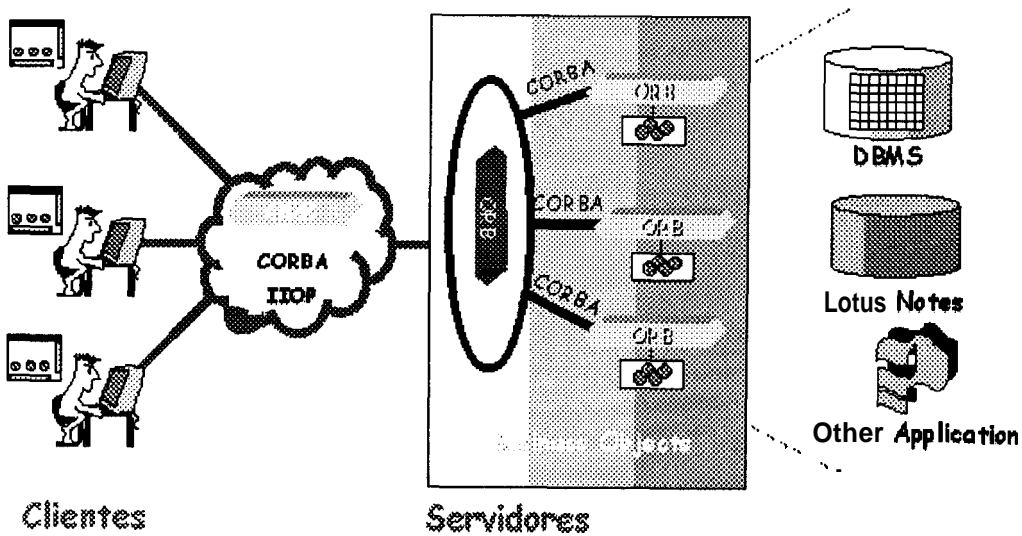


Figura 5. Comunicación a través de IOP (Esquema Three Tier).

Para otros ambientes que no son TCP/IP, se utiliza el protocolo ESIOP (Environment Specific Inter-ORB Protocol).

Los objetos publican sus identidades y ubicaciones utilizando las referencias de objetos. CORBA especifica un formato común para las referencias de objetos llamado IOR (Interoperable Object Reference), el cual contiene perfiles que describen como los clientes pueden encontrar y enviar requerimientos al servidor usando un protocolo particular.

Todos los IORs, deben tener por lo menos un *perfil* IIOP, lo cual asegura que donde sea que se encuentre la referencia de objeto, cualquier ORB que cumpla con CORBA, sera capaz de localizar el servidor y enviarle los requerimientos. El *perfil* IIOP tiene la dirección Internet del servidor y un valor clave usado por el servidor para encontrar el objeto específico descrito por la referencia.

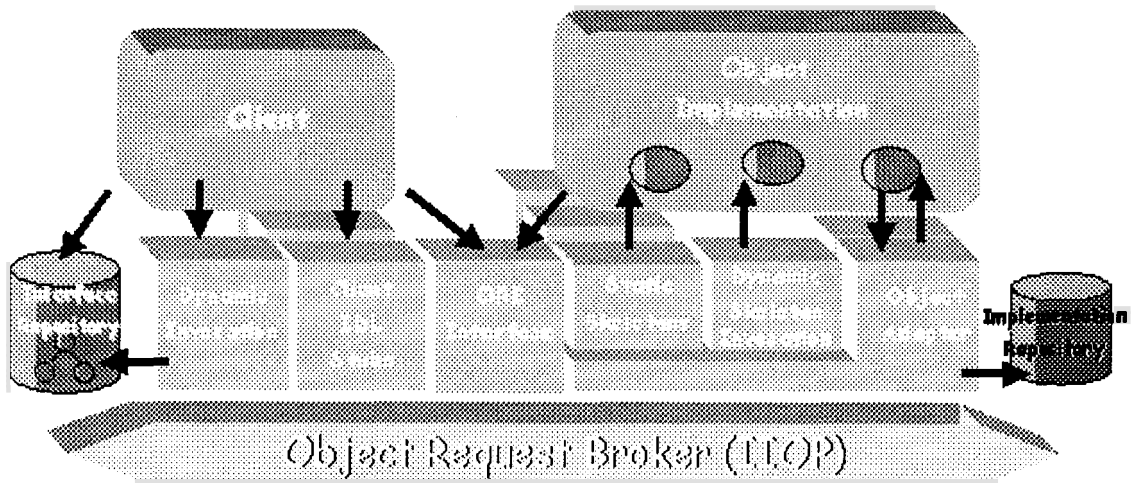
IIOP brinda un ambiente en el cual los programadores definen e invocan cualquier metodo que necesitan, y transparentemente se realiza la comunicacion, es decir, los programadores no tienen que escribir programas IIOP, nunca requieren interactuar con IIOP de ninguna forma, simplemente es invisible para ellos.

1.5.2 ORB (Object Request Broker)

ORB es el camino que establece las relaciones Cliente/Servidor entre los objetos, y abarca todo lo referente a la infraestructura de comunicacion necesaria para localizar e identificar objetos, es decir, permite que un cliente pueda invocar en forma transparente un metodo del servidor, el mismo que puede estar en la misma maquina o en cualquier otra de la red.

ORB intercepta la llamada del cliente y es responsable por encontrar un objeto en el servidor que pueda implementar el requerimiento, le envía los parametros, invoca el método y retorna los resultados al cliente.

En la siguiente figura se indican los componentes principales del ORB.



Estructura de CORBA 2.0 ORB

Figura 6. Estructura de CORBA 2.0 ORB

Lado Cliente

Para hacer un requerimiento, el cliente se comunica con el ORB Core, ya sea a través de la interface de Invocación dinámica (DII) ó del IDL STUB, el mismo que hace creer al cliente que los objetos se encuentran localmente.

Lado Servidor

El ORB Core transfiere el requerimiento al servidor, el mismo que recibe una llamada a través del IDL Skeleton ó del Dynamic Skeleton y se encarga de buscar el objeto capaz de cumplir con el método requerido. Una vez que se ha ejecutado el método, devuelve la respuesta al cliente.

Repositorio de Interfaces

Un repositorio de Interfaces es una base de datos en línea que contiene las definiciones de objetos. Contiene metadata que es idéntica a los componentes que se describen en el IDL. Para organizar y recuperar la información del repositorio, este especifica un conjunto de clases cuyas instancias representan la información que está en el repositorio, es decir, se forma una jerarquía de clases en base a la especificación IDL. Los objetos que conforman el repositorio son versiones compiladas de la información que está en un archivo fuente IDL.

Repositorio de Implementación

Se lo implementa por medio de un archivo que contiene las definiciones de todos los objetos registrados en BOA. Se pueden leer los nombres de instancias de los objetos, las interfaces registradas, los modos de activación del servidor, y los argumentos y variables de ambiente a ser pasadas a cada servidor cuando el mismo ha sido activado.

1.5.2.1 Localizando el ORB

Los metodos de inicializacion de CORBA son los que nos permiten que un Objeto pueda encontrar el ORB, BOA, Repositorio de Interface, Trader, Naming Service, y cualquier cosa que se necesite para que ese objeto forme parte del universo intergaláctico de objetos distribuidos. Estos metodos ORB de proposito general son implementados por el pseudo-objeto `CORBA::ORB`. Un pseudo-objeto es un objeto que el ORB crea, pero que puede ser invocado como cualquier otro objeto. En Java, el pseudo-objeto ORB se refiere a la clase `org.omg.CORBA.ORB`.

Así que, cómo se obtiene una referencia de objeto ORB?. En CORBA 2.0 se hace una llamada al **API** ORB_init para obtener una referencia al pseudo-objeto ORB. En su lugar, en Java, se usa el metodo estatico *init* (o metodo de clase), asumiendo que ya existe una instancia de la clase **org.omg.CORBA.ORB** corriendo en el ambiente. Invocar este metodo estatico es equivalente a hacer la llamada al API ORB-init, ya que ambos retornan una referencia al nuevo objeto ORB que se ha inicializado.

1.5.2.2 Inicialización de un objeto

El servicio de inicializacion ofrece todos los servicios que un objeto necesita para funcionar en un ORB. La Figura 7 muestra las *llamadas* típicas que un objeto debe hacer para formar parte del mundo de objetos distribuidos.

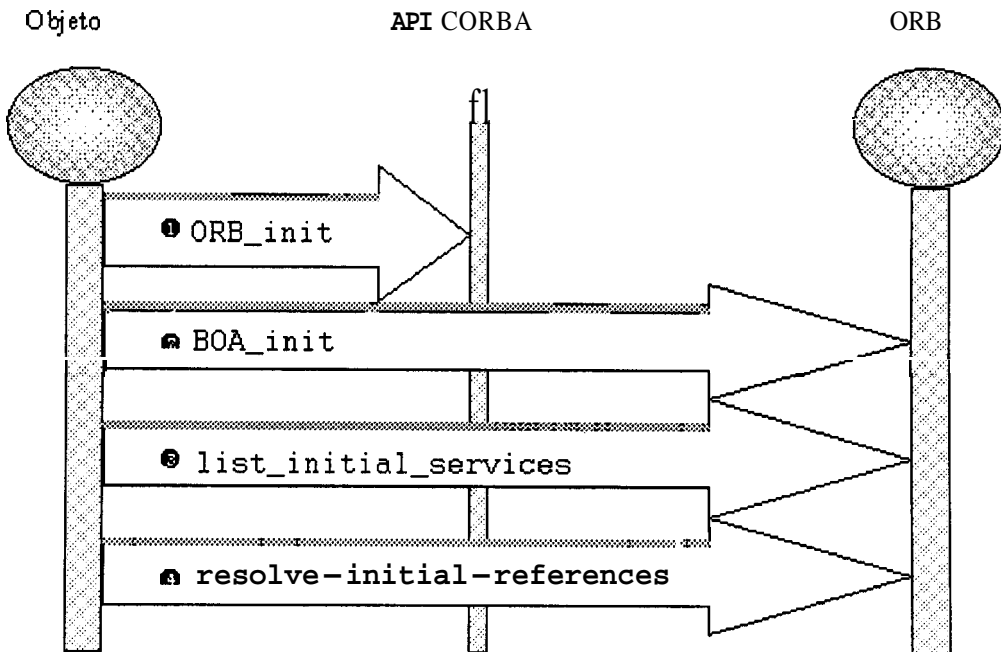


Figura 7. Escenario de inicialización.

A continuación se detalla el escenario de inicialización:

1. **Obtener una referencia al ORB.** Se puede hacer una llamada al API de CORBA `ORB_init`, o en Java invocar el método estático `org.omg.CORBA.ORB.init`, obteniendo con su valor de retorno una referencia a un pseudo-objeto ORB.
2. **Obtener un Puntero al Object Adapter.** Si se usa BOA (Basic Object Adapter, del que se hablara mas adelante), se debe invocar el método `BOA-init`

en el pseudo-objeto ORB (se obtuvo su referencia en el paso anterior) para obtener la referencia de este objeto BOA (BOA también es un pseudo-objeto). Se necesitara esta referencia para registrar sus objetos con el ORB.

3. **Descubrir los objetos iniciales que están disponibles.** Invoque el metodo list-initial-sevices en el pseudo-objeto ORB para obtener una lista de los objetos bien conocidos, e.j. el Repositorio de Interface, Trader, y el Servicio de Nombramiento. Estos objetos bien conocidos, son retornados en una lista de nombres de tipo string.

4. **Obtener las referencias de objeto para los servicios que necesita.** Se invoca el metodo resolve-initial-references para obtener las referencias de objeto de los servicios que se necesita. Luego de estos cuatro pasos se puede considerar un ciudadano de primera clase en el ORB de CORBA.

1.5.2.3 Activación de los objetos

Antes de poder conectarnos a un objeto servidor desde alguna aplicación cliente, este debe haber sido pre-iniciado. Si estamos hablando de millones de objetos servidores, ni la computadora más grande lo soportaría. Para lograr la ilusión que el cliente vea que todos los objetos servidores están activos y corriendo, esperando simplemente ser invocados, se debe proveer al ORB del lado del servidor de una función de startup automática. El ORB debería ser capaz de pre-iniciar un objeto o de iniciarlo bajo demanda cuando el cliente lo invoque. Los objetos servidores junto con el *Basic Object Adapter* (BOA) le dan la ilusión a los clientes que cada objeto servidor que ellos conocen está siempre activo.

El principal propósito de BOA es permitir que el servidor interactúe con el ORB, es decir, se utiliza BOA para decirle al ORB que un objeto está listo para realizar operaciones.

BOA es un pseudo-objeto (creado por el ORB), cuyos métodos son usados para crear o destruir referencias de objetos y para consultar o actualizar la información que BOA mantiene para estas referencias de objeto. BOA mantiene un *registro* de los objetos

activos e implementaciones que controla. Se usa la interface BOA para comunicarse con este *registro* y decirle al ORB acerca de sus objetos.

CORBA 2.0 requiere que el Adaptador BOA este disponible en cada ORB y que ademas incluya las siguientes funciones:

- ◆ Un repositorio de Implementación que deje instalar y registrar alguna implementación de objeto. Debe ademas dejar proveer información de descripción del objeto.
- ◆ Mecanismos para generar e interpretar referencias de objetos, activar y desactivar implementaciones de objetos, e invocar metodos y pasarles sus parametros.
- ◆ Invocaciones de metodos a traves de skeletons.

Pero, ***cómo hago conocer mis objetos servidores?***. En el caso mas sencillo, simplemente se inician todos los objetos cuando se ha activado el proceso en el cual ellos viven. En el caso específico de que, quien active el proceso en el que viven los objetos sea VisiBroker, se debe hacer la llamada al metodo *obj_is_ready* para cada objeto que se inicie y la llamada al metodo *impl_is_ready*, cuando todos los objetos esten listos.

Los objetos permaneceran activos hasta que el proceso servidor termine o hasta que se genere la llamada al metodo *deactivate_obj* para desactivar un objeto particular.

En el caso de que sea una gran cantidad de objetos, los cuales, no puedan permanecer en memoria, VisiBroker ofrece una activacion “just-in-time” que usa una version modificada del metodo *obj-is-ready* para pasar un segundo argumento que especifique un “Activador” para ese objeto que se quiere activar.

VisiBroker provee un Repositorio de Implementacion Distribuido llamada “impl_rep” ubicado en el directorio donde el ORB esta instalado. Este archivo contiene definiciones de implementacion para todos los objetos registrados con BOA.

Se puede usar el comando “listimpl” para leer el contenido de un Repositorio de Implementacion. Se pueden listar todas las interfaces registradas, nombres de instancias de objetos, la ruta de cada ejecutable de implementacion, modos de activacion del servidor, y los argumentos y variables de ambiente a ser pasadas a cada servidor cuando este activado.

Capítulo 2

DESCRIPCION DEL PROYECTO

2.1 Objetivos

- Crear un sistema basado en el esquema Cliente/Servidor, que modele principalmente la reservación de vehiculos de una oficina de alquiler de autos, contemplando el esquema “Multi-tier”.
- Demostrar la gran aplicabilidad de la arquitectura CORBA, utilizada para la funcionalidad de nuestro proyecto.
- Explotar la facilidad en la obtencion de recursos distribuidos a traves de Internet.

2.2 Justificación

En la actualidad la gran acogida de Internet hacia los negocios ha permitido que nuevas aplicaciones sean desarrolladas reinventando la manera de hacer negocios. Razon por la

cual hemos desarrollado un sistema que ayude a los empleados de una compañía de alquiler de vehículos y que facilite a sus clientes rentar un vehículo de acuerdo a sus necesidades y gustos, además de, mostrar información actualizada y ofrecer un servicio de calidad.

En definitiva, el desarrollo de este Sistema nos permitira determinar si el Web de Objetos esta listo para formar parte del lucrativo mercado de las transacciones en linea.

2.3 Especificaciones del proyecto

La aplicacion permite la reservacion de un vehiculo de una oficina Rent-a-car, para lo cual, se implementaron las siguientes transacciones principales:

- **Reservación de un vehículo:** Se especifica a nombre de quien se hace la reservacion, el tipo de vehiculo, y el numero de dias que se lo usara. La transacción devuelve la tarifa total a cancelar.
- **Cancelación de reservaciones:** Respectivos vehiculos vuelven a estar disponibles.
- **Entrega del vehiculo al cliente:** La entrega se la realiza habiendose realizado previamente una reservacion.

- **Devolución del vehículo:** Al hacer la devolución, los vehículos vuelven a quedar en estado disponible. El Monto a cargar en la tarjeta de crédito sera de acuerdo al numero de dias y de horas que realmente se utilizo el vehiculo, y por supuesto, de acuerdo al **tipo** de vehiculo usado (cada vehiculo tendra su propia tarifa).

La empresa inicia sus operaciones con cierta cantidad de vehiculos con su respectiva tarifa (datos iniciales que están pregrabados en la Base de datos, y de no ser asi, se puede utilizar la misma aplicacion para realizar el ingreso).

El Sistema cuenta además con transacciones que permiten: *Ingresar nuevos vehiculos*, *Dar de baja vehiculos* o *Editar vehiculos ya ingresados*. Estas permitirian respectivamente, ingresar nuevos vehiculos para su alquiler (con su respectiva tarifa), darlos de baja y cambiar los datos ya ingresados. La empresa tiene tarifas distintas en dias ordinarios y en fines de semana, tarifas de 1 dia, de 3 dias, o de toda una semana. Se incluye Seguros (no obligatorios) para el vehiculo.

2.4 Descripción general del proyecto

El Sistema de Alquiler de Vehiculos esta desarrollado bajo el esquema Cliente/Servidor, utilizando la arquitectura CORBA, JDBC, y JavaBeans. La idea es crear una aplicacion cliente/servidor 3-capas completa, destinada para ser usada en el Web usando CORBA y JavaBeans. En sí, este Sistema se presenta en forma de Applets que son invocados desde una pagina HTML, la cual, es cargada desde algún Web Server.

En esta aplicacion 3-capas cliente/servidor, los Applets clientes invocan operaciones en objetos servidores CORBA, de la capa intermedia, a traves de un IIOP ORB. Los objetos servidores proveen la lógica del negocio y almacenan sus datos persistentes en una base de datos SQL que soporta JDBC.

Las páginas HTML y Applets residen en la máquina donde se encuentra el Web Server, mientras que los metodos o funciones remotas de los objetos CORBA, que son invocados por los Applets, pueden residir en la misma maquina que hace de Web Server o en alguna otra maquina de la red.

2.4.1 Detalles del Cliente

Una aplicación cliente/servidor es por su naturaleza manejada por el lado cliente. Un Applet de Java logra realizar esto manipulando beans visuales. Cualquier cosa que el usuario haga con estos beans se convierte en una fuente de invocación de métodos CORBA. Esto significa que el objeto servidor espera en forma pasiva algún requerimiento de algún cliente para lanzar la ejecución de sus métodos.

El cliente está implementado con Applets de Java. El Applet cargado junto con la página HTML realiza funciones básicas como:

- ◆ Validación de datos ingresados.
- ◆ Invocación de funciones en el lado del servidor para que ejecuten cierto procesamiento con los datos ingresados y con los datos persistentes de la base.
- ◆ Presentación gráfica del resultado devuelto por las funciones o métodos del servidor.

2.4.2 Detalles del Servidor

En la tercera capa, los beans clientes COMA se comunican con los objetos servidores COMA. Los objetos servidores a su vez, se comunican con uno o mas DBMSs por medio de JDBC.

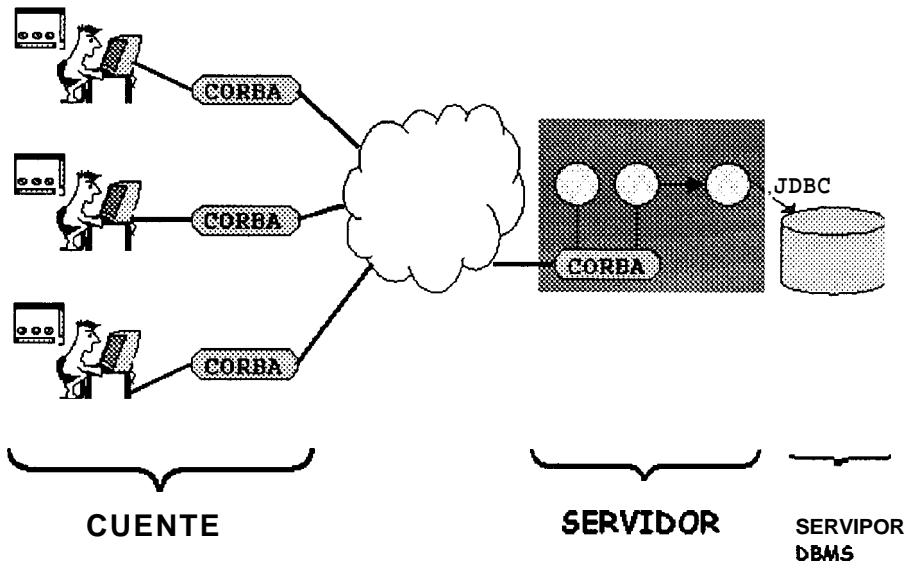


Figura 8. Esquema de 3 capas con CORBA

En la figura anterior, se puede observar cada una de las capas que conforman el Sistema, notándose que la tercera capa consiste de la base de datos JDBC, que es donde se almacena el estado persistente de los objetos.

El programa servidor del Sistema fue desarrollado en lenguaje Java y utilizando las ventajas ofrecidas por la arquitectura CORBA. Consiste basicamente de metodos o fnciones de Objetos CORBA que pueden ser invocados desde un Applet, el cual, puede ejecutarse en cualquier Browser que haya cargado previamente la respectiva pagina HTML.

Dicha invocacion de fnciones está embebida en el Applet y realizan acciones en el lado del servidor tales como:

- ◆ Levantar conexiones a la Base de datos para cada cliente, y
- ◆ Realizar operaciones y calculos específicos sobre los datos de la base.

Como se utiliza invocacion estatica, el ORB obtiene la definición de los objetos con los que necesita trabajar de los fragmentos de codigo, tanto del lado cliente como servidor, es decir, no utilizamos el Repositorio de Interfaces, destinado para invocaciones dinamicas.

2.4.3 Procesos involucrados en el Sistema

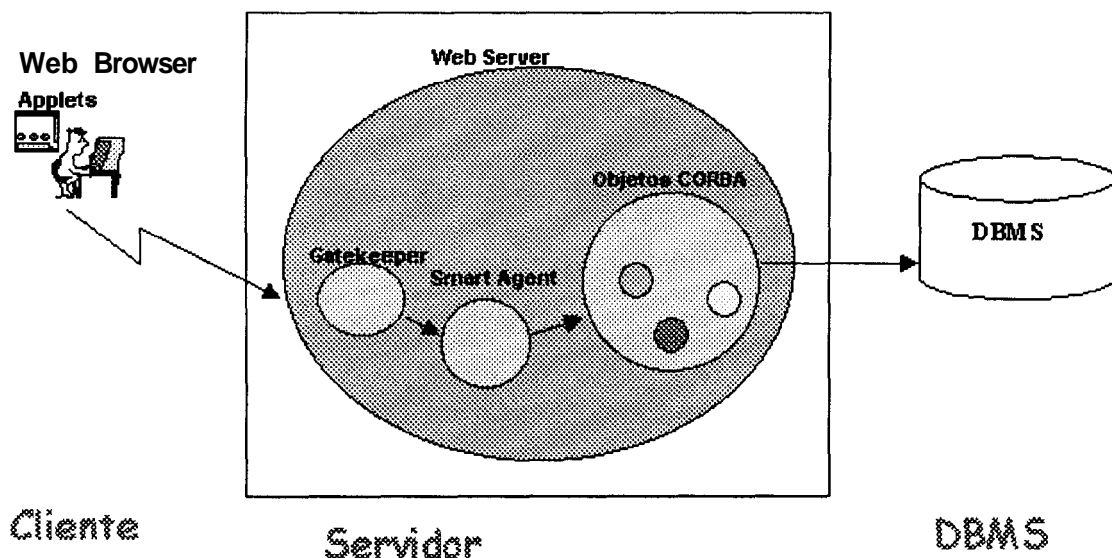


Figura 9. Procesos involucrados en el Sistema

Como se observa en la figura 9, del lado cliente tenemos como procesos a los *Applets de Java* y al *web browser* necesario para que se ejecuten.

Del lado Servidor, los procesos que interactúan como un conjunto para atender requerimientos de clientes, *en este sistema*, son: el Web Server, el Smart Agent, el Gatekeeper, y el proceso servidor en Java que alberga los objetos CORBA activados.

El *Web Server* no hace mas que atender los requerimientos HTTP, permitiendo descargar las páginas HTML a los web browser clientes, asi como el Applet enlazado en la pagina que, en el caso de este sistema, permitira la interacción Cliente/Servidor.

El *Smart Agent* permite localizar la implementación especificada (objeto CORBA), cuando una aplicacion cliente invoque el método "bind" en un objeto, de tal forma que se pueda establecer la conexión entre el cliente y ese objeto. Para localizar el Smart Agent las aplicaciones cliente envian un mensaje broadcast , y el primer Smart Agent en responder seria usado. Una vez que el Smart Agent ha sido localizado, se establece una comunicacion UDP punto-a-punto entre el cliente y el objeto. Al implementar este Sistema con Applets, todos los mensajes broadcast van a ser manejados por el Gatekeeper, al cual estos Applets están conectados.

El *Gatekeeper* ayuda a sobrellevar algunas de las restricciones de seguridad impuesta por los web browsers. Realiza las veces de un proxy, puesto que recibe las invocaciones de objetos por parte de los clientes, para luego hacerle un forward de este requerimiento al objeto específico. Permite además, invocar objetos que no residen en el web server, recibir callbacks, y ser usado como un demonio HTTP, eliminando la necesidad de usar un servidor HTTP durante la fase de desarrollo de la aplicación.

Finalmente, el *proceso servidor en Java* simplemente contiene las instancias de los objetos CORBA activos a ser invocados por los clientes.

2.4.4 Descripción de la Persistencia del proyecto

La persistencia es un modelo de clases, el mismo permite que las instancias de clases se puedan almacenar en una base de datos, que para nuestro caso es una base de datos relacional. *Este modelo permite que cada objeto sepa guardarse independientemente en la base de datos.*

Este modelo inicialmente se conecta a la Base de Datos, luego traduce el comportamiento de un objeto y lo lleva a sentencias SQL por medio de un conjunto de metodos que mapean uno a uno los atributos de la clase a columnas de una tabla.

Generalmente se mapea una clase a una tabla, pero esto depende del performance que debe tener el diseño del modelo de clases del negocio, ya que se debe tomar en cuenta que a medida que crece el numero de objetos, crece el numero de tablas y el sistema

aumenta en complejidad ya que es mas difícil y lento obtener la información que se encuentra en la base de datos.

Para que un objeto sea persistente, el mismo debe heredar de la clase Persistencia, es decir, este objeto adquiere el comportamiento de la clase, con lo cual aprende a levantarse de la base de datos, guardarse, actualizarse, mantener estados, etc.

Capítulo 3

ANÁLISIS Y DISEÑO DEL SISTEMA

3.1 Introducción

El análisis y diseño del sistema fue realizado con la metodología de Orientación a objetos, utilizando una combinación de las técnicas de OMT/Rumbaugh, Booch y Jacobson.

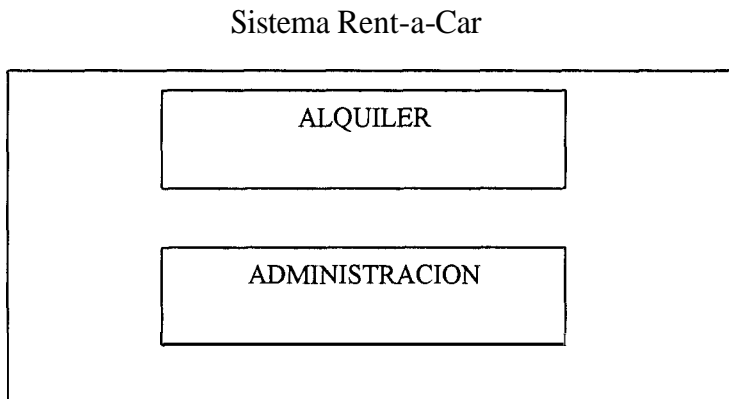
Al emplear esta metodología se logra una abstracción precisa de que debe hacer el sistema, para luego en el diseño tomar decisiones de alto nivel sobre la funcionalidad general del sistema.

Empezaremos haciendo un levantamiento de los requerimientos del Sistema (Capítulo 2), que son las necesidades de los usuarios del mismo, para lograr un listado de objetos, casos de uso y diagramas que muestren gráficamente la interacción entre los componentes del sistema.

3.2 Analisis de requerimientos

Para el Analisis y Diseño del Sistema de Alquiler de autos, se ha determinado lo siguiente:

Limites del Sistema



Alquiler

Descripción: En este proceso se registran los movimientos realizados en la Empresa, los cuales incluyen: Reservación de vehiculos y Cancelacion de una reservacion.

Actor: Cliente

Empleado de la empresa

Base de datos de Vehiculos

Administración

Descripción: En este proceso, se muestran las opciones para entrega de vehículos a los clientes, devolución de vehículos por parte de los clientes, ingresar nuevos vehículos, dar de baja un vehículo existente, editar un vehículo existente y cambiar password del administrador.

Actor: Administrador

Base de datos de vehículos

3.3 Casos de Uso

3.3.1 Lista de casos de Uso

3.3.1.1.- Alquiler

- 1) Reservación de vehículos
- 2) Cancelación de reservación

3.3.1.2.- Administración

- 1) Entrega de vehículo
- 2) Devolución del vehículo
- 3) Ingreso de nuevos vehículos
- 4) Dar de baja a vehículos dañados

- 5) Modificar vehiculo existente
- 6) Cambiar password de administrador

3.3.2 Documentación de casos de uso

3.3.2.1.- Alquiler

Reservación de vehículos

Descripción	Cliente se acerca a la oficina a reservar un vehiculo, especifica dias y/o horas de alquiler, tipo de vehiculo y finalmente el vehiculo que desea reservar de acuerdo al tipo.
Nota:	<ul style="list-style-type: none"> ◆ Si el auto solicitado no se encuentra disponible, el cliente debe buscar otro vehiculo de su agrado. ◆ El sistema NO verifica problemas en cupos de tarjetas de credito. ◆ Se indica al cliente el total a pagar. ◆ Se permite ingresar datos de cliente, en caso de no existir.
Valor Medible:	◆ Reservación de vehiculo registrado.

Cancelación de la reservación de vehículo

Descripción:	Cliente se acerca a la oficina de alquiler y manifiesta que ya no desea alquilar el vehículo.
Nota:	◆ El vehículo queda disponible. Y si no ha pasado la fecha de reservación, se cancela la misma
Valor Medible:	◆ Reservación Cancelada.

3.3.2.2. Administración

Entrega de vehículo

Descripción	Cuando el cliente se acerca a la oficina de alquiler, ya sea para reservar un vehículo y retirarlo o solamente a retirarlo gracias a una reservación previa vía Web.
Notas:	◆ El sistema verifica fecha de reservación.
Valor Medible:	◆ Vehículo entregado.

Devolución de vehículo

Descripción:	Cliente se acerca a devolver el vehículo.
Notas:	<ul style="list-style-type: none"> ◆ El sistema calcula valor a cargar en base a los días y horas ocupados. ◆ El vehículo vuelve a quedar disponible.
Valor Medible:	<ul style="list-style-type: none"> ◆ Disponibilidad del vehículo. ◆ Cliente satisfecho con el servicio.

Ingreso de nuevo vehículo

Descripción:	Administrador ingresa al Sistema e ingresa un nuevo vehículo
Notas:	<ul style="list-style-type: none"> ◆ En el sistema se ingresan todos los datos del vehículo como nombre, marca, modelo, año, color, estado, nombre gráfico asociado al vehículo. ◆ El vehículo queda disponible para el alquiler.
Valor Medible:	<ul style="list-style-type: none"> ◆ Disponibilidad del vehículo. ◆ Cliente satisfecho con el servicio

Dar de baja a un vehículo

Descripción:	Administrador ingresa al Sistema y decide dar de baja a vehículos que se encuentran en malas condiciones para el alquiler.
Notas:	◆ En el sistema se encuentra registrado el vehículo.
Valor Medible:	<ul style="list-style-type: none"> ◆ Vehículo no existe en el sistema. ◆ Cliente satisfecho con el servicio porque siempre se renueva el parque automotor.

Cambiar password de Administrador

Descripción:	Administrador ingresa al Sistema y debe cambiar su password por seguridad en el Sistema.
Notas:	◆ En el sistema se encuentra registrado el password del administrador, el mismo que se utiliza para las operaciones de entrega de vehículo,.
Valor Medible:	◆ Satisfacción administrativa por la seguridad del sistema.

3.4 Lista de Objetos

El primer paso para construir un modelo de objetos es identificar clases de objetos relevantes del dominio de la aplicación (Figura 6). Los objetos incluyen entidades físicas, como casas, empleados y maquinas, y los conceptos como trayectorias, calendarios de pago, asignaciones. Todas las clases de objetos deben tener sentido dentro del dominio de la aplicación, para evitar construir implementaciones innecesarias como listas enlazadas o subrutinas.

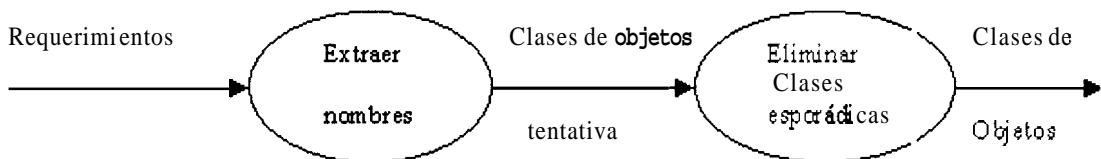


Figura 6. Identificando clases de objetos

En el sistema de Alquiler de Autos encontramos las siguientes clases:

Reservación	Factura
Vehiculo	Devolución
Oficina	Entrega
Transacción	Disponibilidad
Tarifa	Base de datos
Pago	
Cliente	

Para obtener el listado definitivo, se eliminan de acuerdo a los siguientes criterios:

Clases redundantes.- Si dos clases expresan la misma información, el nombre mas descriptivo debe mantenerse.

Clases irrelevantes.- Si alguna clase tiene muy poco o nada que hacer con el problema, debe ser eliminada. Se debe considerar todos los contextos donde la clase puede ser importante.

Clases Vanas.- Una clase debe ser específica.

Una vez realizado el análisis respectivo, los objetos definitivos son:

Reservación

Vehiculo

Tarifa

Pago

Cliente

3.4 Definición de Escenarios

3.4.1 Alquiler

Escenarios del caso de Uso: Reservación de un Vehículo

3.4.1.1 Reservación Exitosa de un vehículo

3.4.1.2 Reservación de un vehículo ya reservado en una fecha específica

Escenario **3.4.1.1** Reservación Exitosa de un Vehículo.

Asunciones:

1. El vehículo está disponible en las fechas que desea reservar el cliente.
2. Los datos del cliente están correctos.
3. El cliente tiene crédito.

Resultados:

1. Se crea una reservación con toda la información ingresada del cliente
2. El vehículo se reserva en las fechas ingresadas en la reservación

Objetos Involucrados:

Reservación
Pago
Cliente
Vehículo
Empleado

Escenario 3.4.1.2 Reservación de Vehículo ya reservado en una fecha específica

Asunciones:

1. Vehículo se encuentra reservado en una fecha determinada

Resultados:

1. Reservación no exitosa
2. Vehículo no se reserva

Objetos Involucrados:

Reservación
Cliente
Vehículo
Empleado

Escenarios del caso de Uso: Cancelación de reserva de un Vehículo

3.4.1.3 Cancelacion Exitosa de un vehiculo

3.4.1.4 Cancelacion de un vehiculo entregado

3.4.1.5 Cancelacion de una reservacion pasada la fecha de reservacion

Escenario 3.4.1.3: Cancelación Exitosa de un vehículo

Asunciones:

1. La reservacion que se desea cancelar, debe existir.
2. El vehiculo que se ha reservado no debe haber sido entregado.
3. La fecha de cancelacion debe ser menor o igual a la fecha de reservacion del vehiculo.

Resultados:

1. Se cancela la reservacion

Objetos Involucrados:

Reservación
Empleado

Escenario 3.4.1.4: Cancelación de un vehículo entregado**Asunciones:**

1. La reservacion que se desea cancelar debe existir.
2. El vehiculo ha sido entregado.

Resultados:

1. No se cancela la reservación

Objetos Involucrados:

Reservación
Empleado

Escenario 3.4.1.5: Cancelacion de una reservacion pasada la fecha de reservacion.

Asunciones:

1. La reservacion que se desea cancelar debe existir.
2. La fecha en la que se quiere cancelar es mayor a la fecha de reservacion del vehiculo.

Resultados:

1. No se cancela la reservacion

Objetos Involucrados:

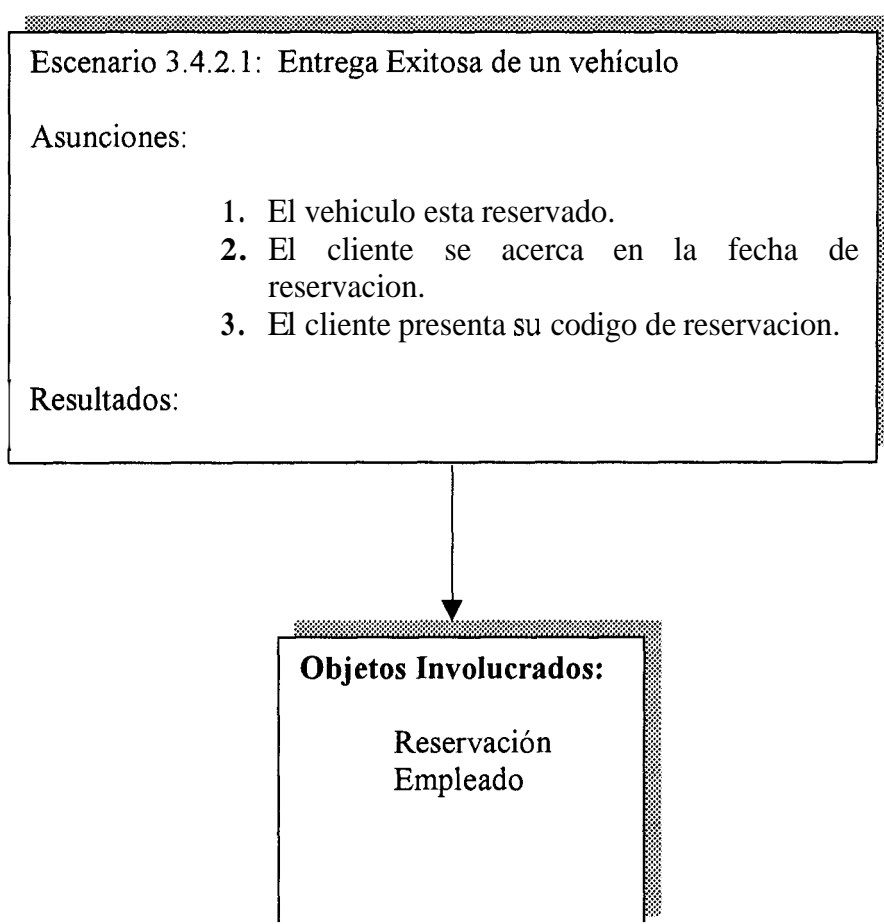
Reservación
Empleado

3.4.2 Administración

Escenarios del caso de Uso: Entrega de un Vehículo

3.4.2.1 Entrega Exitosa de un vehiculo

3.4.2.2 Entrega de un vehiculo pasada la fecha de reservacion



Escenario **3.4.2.2**: Entrega de un vehiculo pasada la fecha de reservación

Asunciones:

1. El vehiculo esta reservado.
2. El cliente se acerca en una fecha que no corresponde a la fecha en que lo reservó.

Resultados:

1. El vehiculo no es entregado al cliente.

Objetos Involucrados:

Reservación
Empleado

Escenarios del caso de Uso: Devolución de un Vehículo

3.4.2.3 Devolución Exitosa de un vehículo

Escenario 3.4.2.3: Devolución Exitosa de un vehículo

Asunciones:

1. El vehículo está reservado.
2. El vehículo fue entregado al cliente.

Resultados:

1. El vehículo queda disponible para alquilar.
2. Se presenta un mensaje por el valor a cargar equivalente al tiempo usado.

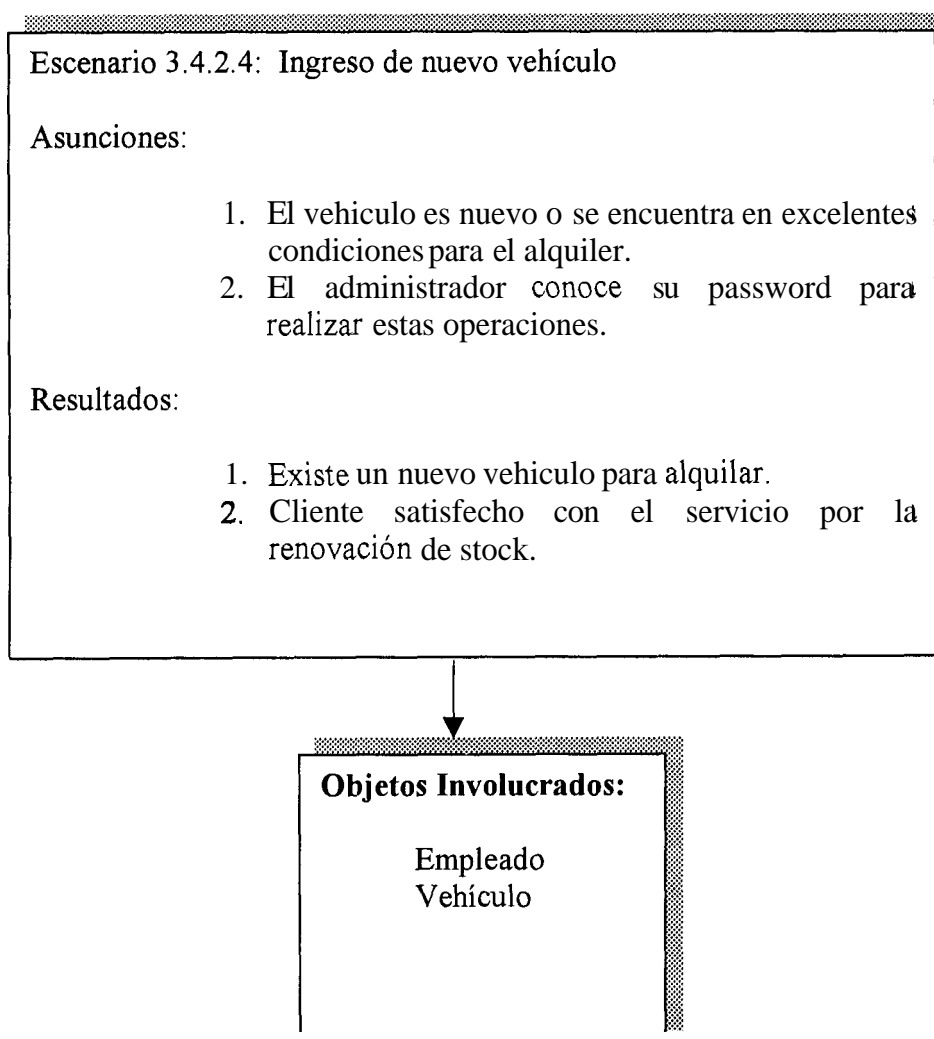
Objetos Involucrados:

Reservación
Pago
Empleado

Escenarios del caso de Uso: Mantenimiento de Vehículos

3.4.2.4 Ingreso de nuevo vehículo

3.4.2.5 Dar de baja a vehículo existente



Escenario 3.4.2.5: Dar de baja a un vehículo existente**Asunciones:**

1. El vehículo se encuentra en malas condiciones para el alquiler del mismo.
2. El administrador conoce su password.

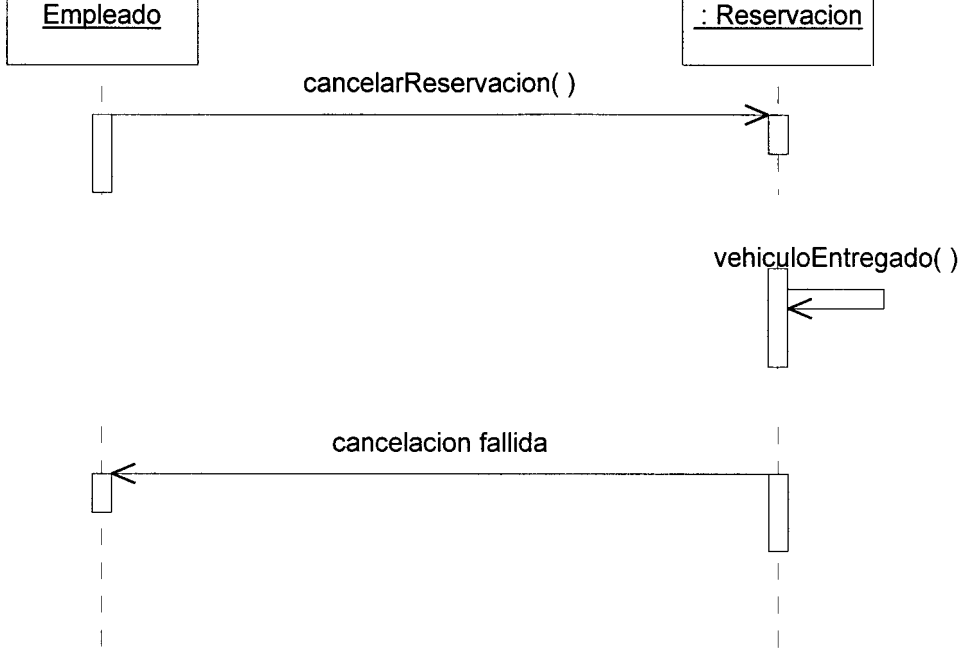
Resultados:

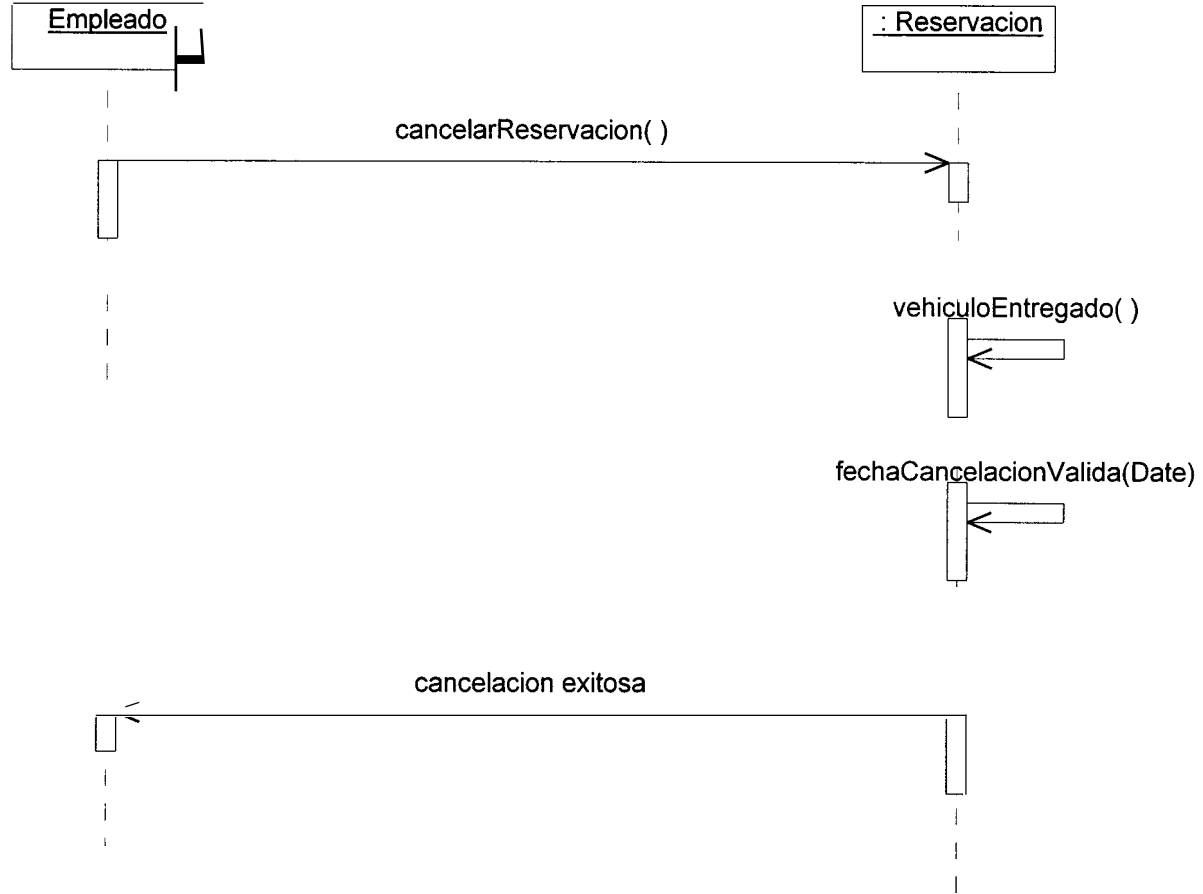
1. Vehículo no se encuentra disponible.

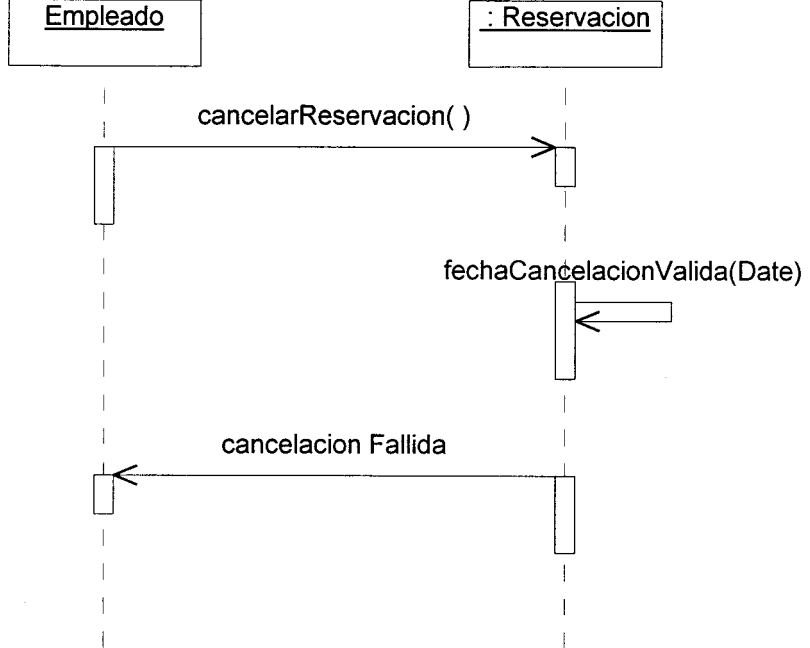
Objetos Involucrados:

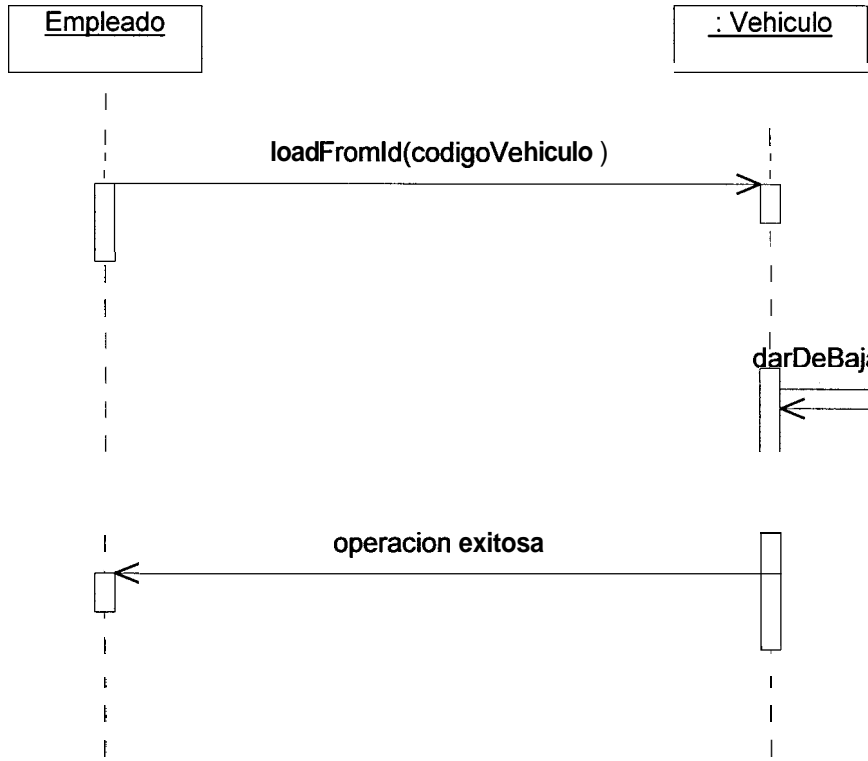
Empleado
Vehículo

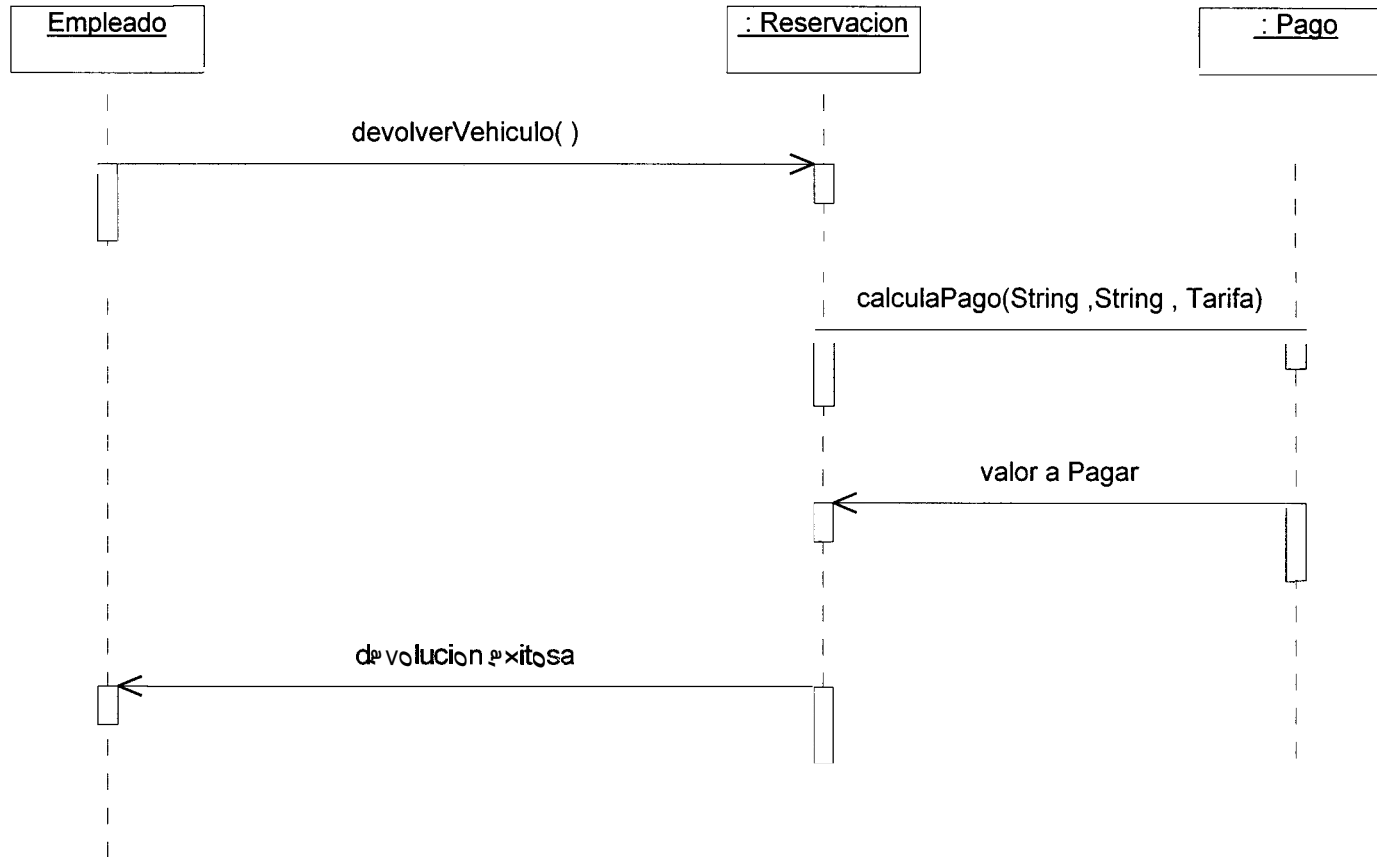
3.5 Diagrama de Interacción de Objetos (DIOs)

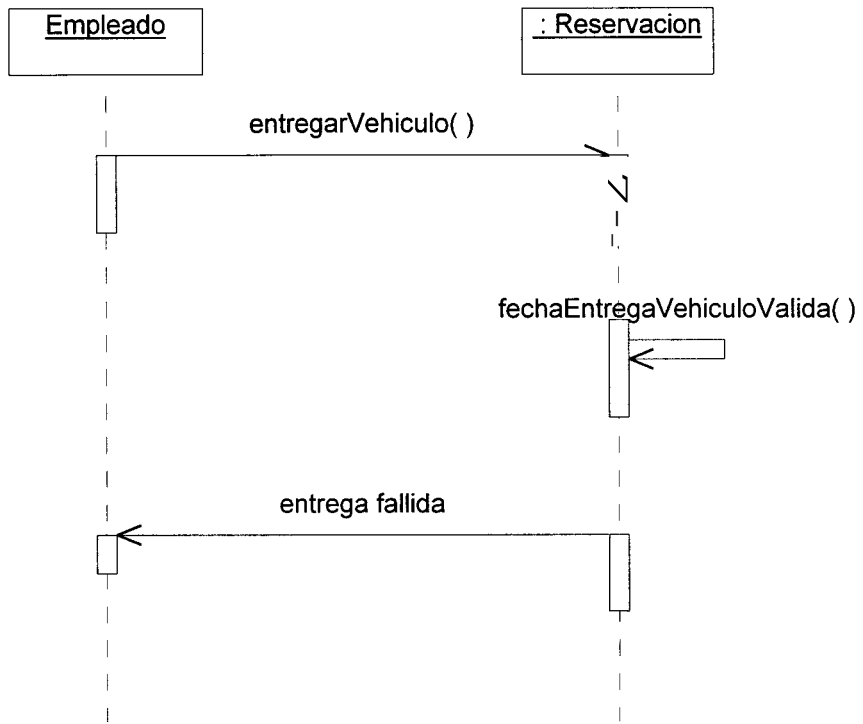






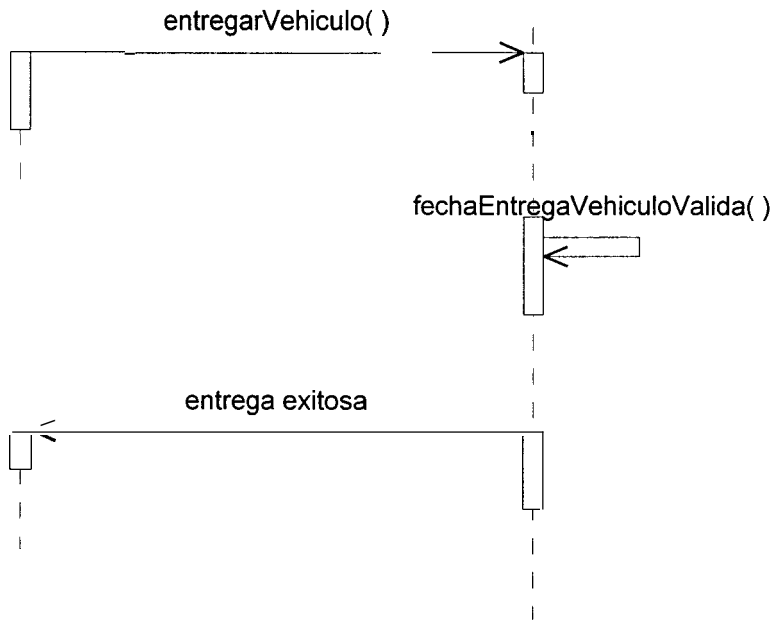


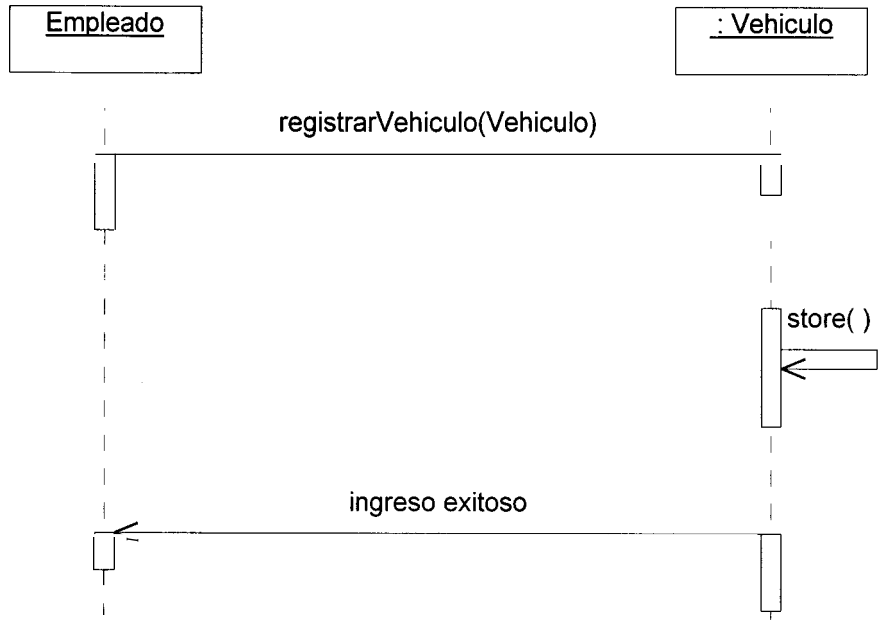


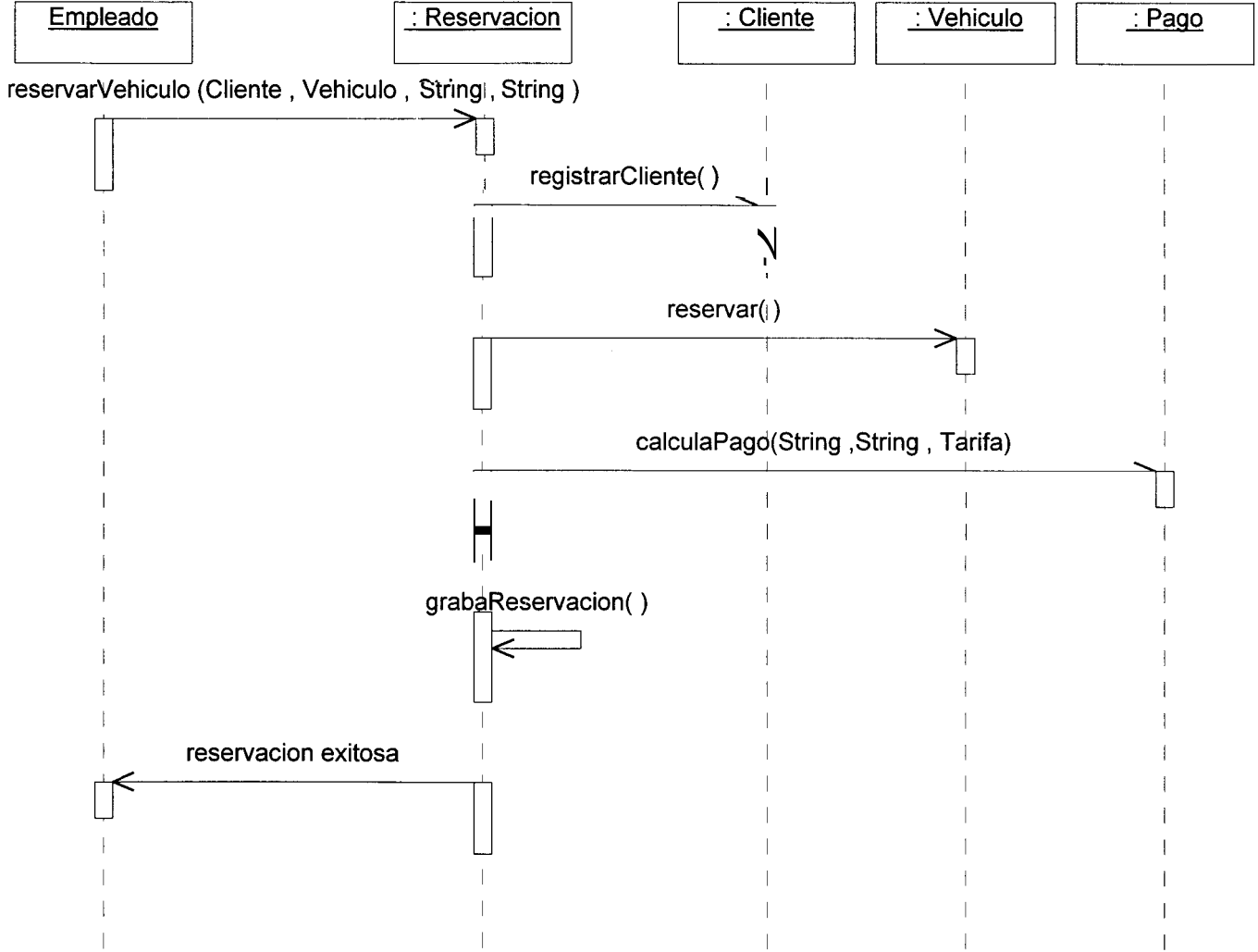


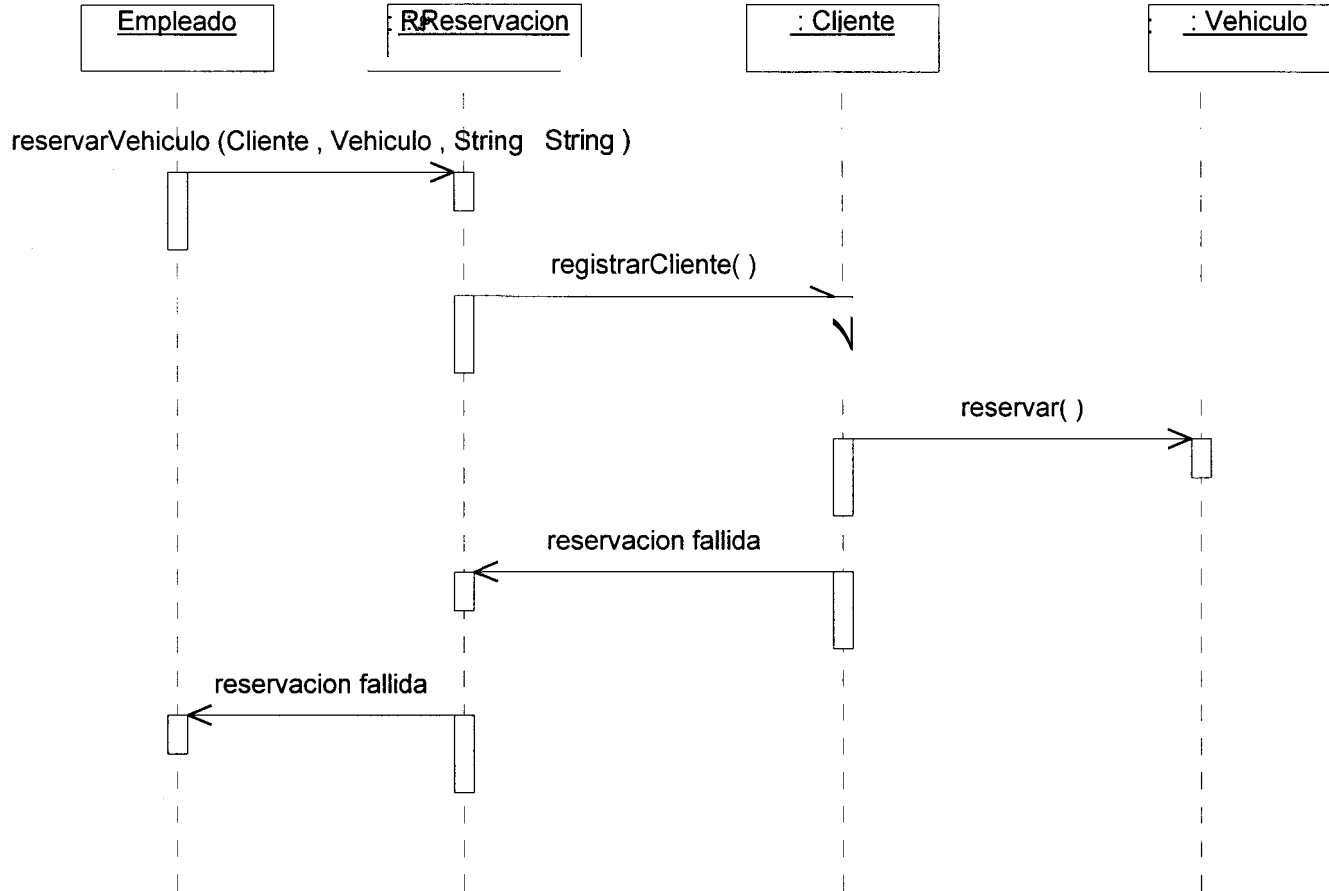
Empleado

: Reservacion

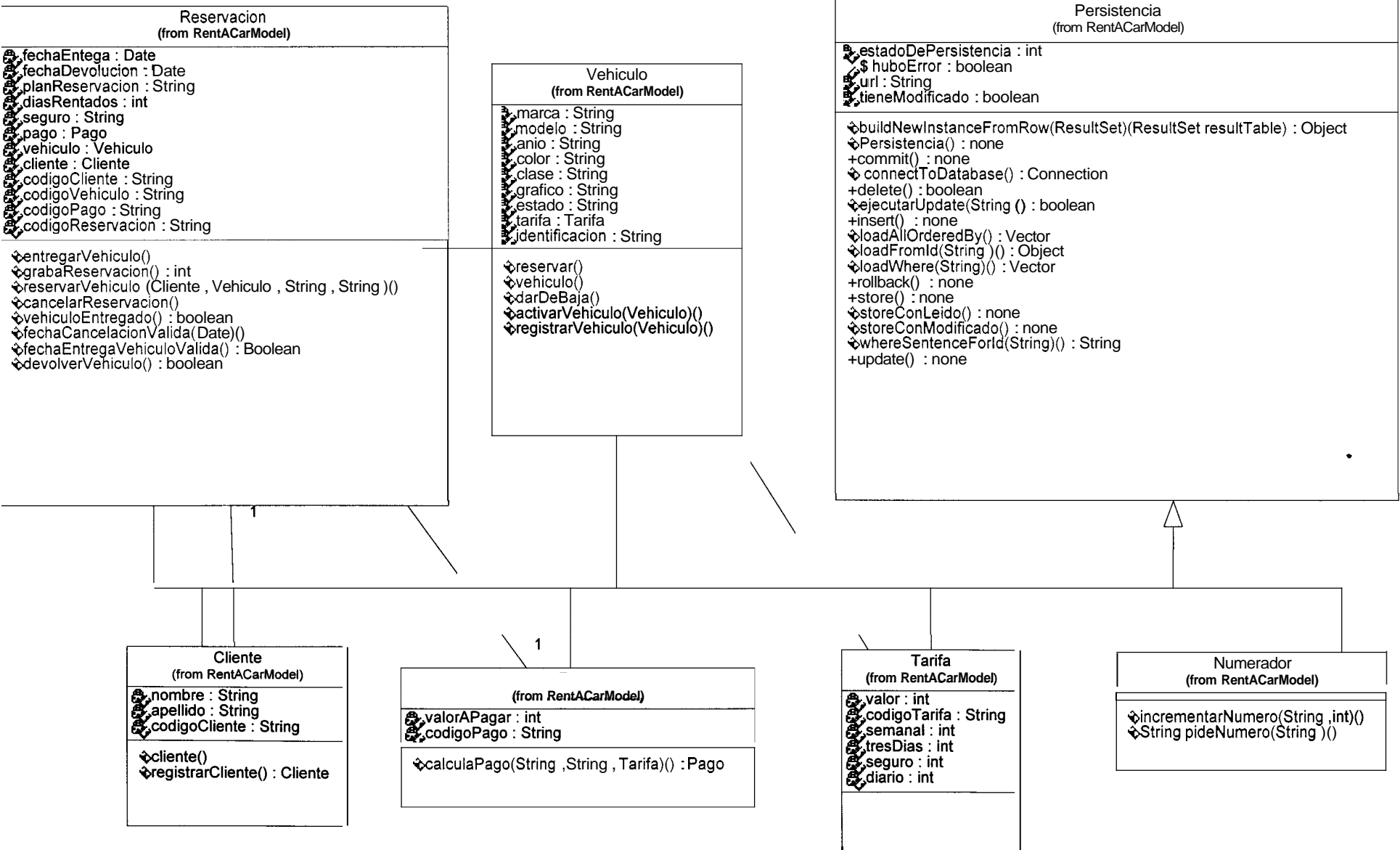




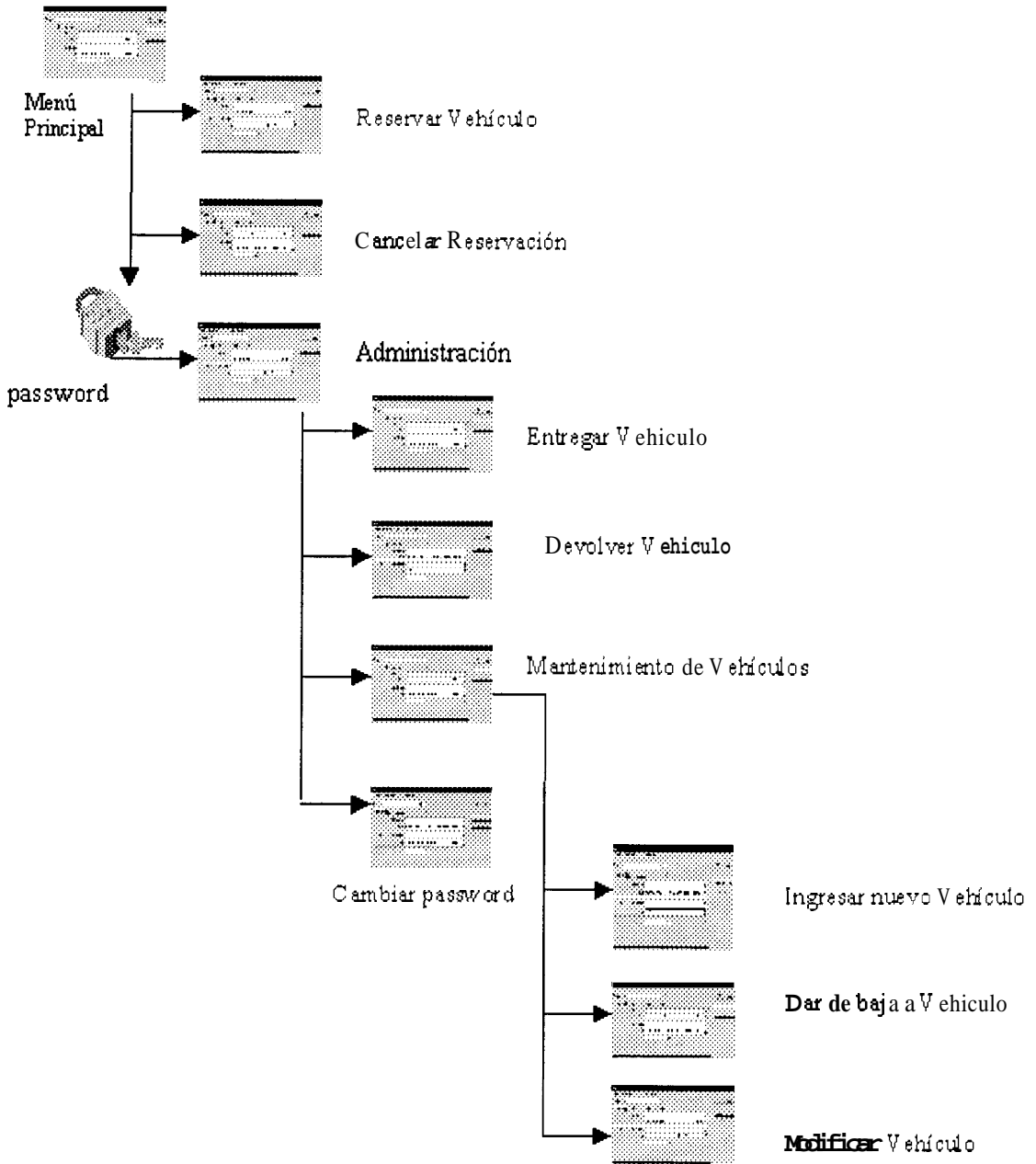




3.6 Modelo de Clases

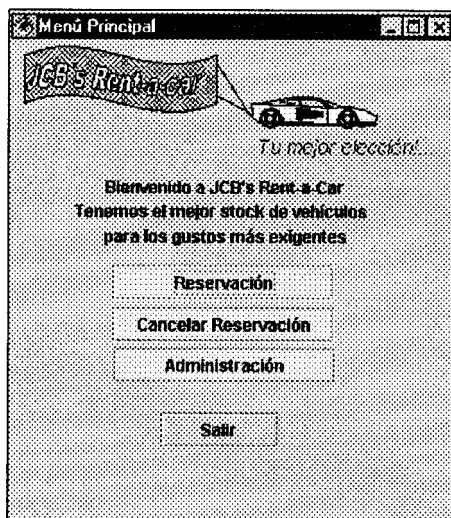


3.7 Flujo de ventanas



3.8 Layouts de ventanas

A continuación se mostrara el diseño de las pantallas del Sistema.



Menu Principal



Selección del Tipo de Vehículo

Datos del Vehículo

Tu mejor elección!...

Información del Vehículo:

Código: 16
 Marca: Chrysler
 Modelo: Cherokee
 Año: 1999
 Color: Plateado
 Clase: Completo

Selecciona un Vehículo

Datos del vehiculo a reservar por el cliente

Reservación de Vehículos

Tu mejor elección!...

Datos Cuenta:

Nombre:
 Apellidos:
 Tarj. de Cred. #:

Fecha de Reservación:

Fecha: mm dd aaaa
 Hora: hh mm AM

Fecha de Devolución:

Fecha: mm dd aaaa
 Hora: hh mm AM

Incluir Seguro

Tarifa				
Diaria:	Tres Dias:	Semanal:	Hora:	Seguro:
\$ 230	\$ 490	\$ 1030	\$ 20	\$ 80

Reservación de Vehiculo

Entregar Vehículo

JCB's Rent-a-car

Tu mejor elección!...

Código de Reserva: 61

Datos del Reservación:

Nombre: Melde

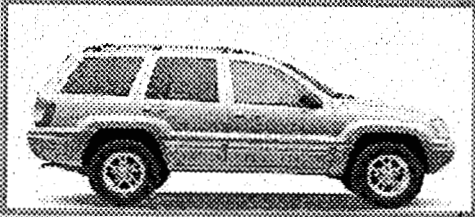
Apellido: Gomez

Tarj. de Cred. #: 579896

Fecha de Reserva:

Fecha: 07/14/1999 (dd/mm/aa)

Hora: 02:15 PM



Entrega del vehiculo al cliente

Cambio de Password

JCB's Rent-a-car

Tu mejor elección!..

Password Antigua:

Password Nuevo:

Confirmar Nuevo:

Cambiar password de Administrador

3.9 Plan de pruebas

Nombre del Grupo de prueba: Reservar Vehículo

Numero de prueba: 1.1

Pre-requisitos:

Enrique desea reservar un vehiculo, tiene buen crédito y no hay problemas con su Tarjeta de credito.

Instrucciones de configuración:

- ◆ El Sistema de Alquiler de Autos, tiene el vehiculo requerido disponible.
- ◆ Enrique ingresa correctamente sus datos personales en el applet de la pagina Web de la compañía.
- ◆ Enrique cumple con todos los requisitos.

Instrucciones de Prueba:

Enrique ingresa al menu principal del Sistema de Alquiler de Autos y previa selección del tipo de vehiculo, se procede a la reserva de un vehiculo.

Comportamiento aceptable:

- ◆ El sistema ingresa a la base de datos y cambia el estado del vehiculo a reservado.

- ◆ El sistema muestra un código de reservación, que el cliente usará para ir a retirar el vehículo de las oficinas.
- ◆ Se procede a mostrar el total de la reservación a cargar en la tarjeta de crédito.

Nombre del Grupo de prueba: Cancelar Reservación

Numero de prueba: 1.2

Pre-requisitos:

Enrique es un cliente de la Compañía, su código de reservación es **123**, y desea cancelar la reservación.

Instrucciones de Configuración:

- ◆ La Compañía de alquiler de autos, tiene 100 clientes incluyendo a Enrique, el Sistema tiene toda la información de Enrique.
- ◆ Enrique solo puede cancelar la reservación hasta la fecha inicial de la reservación.

Instrucciones de Prueba:

- ◆ Enrique ingresa al menú principal y escoge la opción de Cancelar Reservación.

- ◆ Enrique ingresa su código de reservación y procede a cancelar la reservación.

Comportamiento Aceptable:

- ◆ El sistema actualiza la base de datos, cambiando el estado del vehículo de reservado a disponible.
- ◆ El sistema muestra un mensaje indicando que se ha cancelado la reservación.

Nombre del Grupo de prueba: Entregar Vehículo

Numero de prueba: 2.1

Pre-requisitos:

Enrique llega hasta la Compañía de Alquiler de autos, para que le entreguen el vehículo, en la fecha de inicio de reservación. El administrador conoce su password. Enrique conoce su código de reservación.

Instrucciones de Configuración:

- ◆ El vehículo cumple con las especificaciones requeridas por Enrique.

Instrucciones de Prueba:

- ◆ El administrador ingresa su password y se muestra la opción de Entregar Vehículo.

- ◆ Enrique le indica su código de reservación.

Comportamiento Aceptable:

- ◆ El sistema registra la entrega del vehículo cambiando el estado del vehículo a entregado.

Nombre del Grupo de prueba: Devolver Vehículo

Numero de prueba: 2.2

Pre-requisitos:

Enrique desea devolver el vehículo en la fecha final de reservación. El administrador conoce su password.

Instrucciones de Prueba:

- ◆ El administrador ingresa a la opción Devolver Vehículo e ingresa su password.
- ◆ Se procede a revisar el estado del vehículo entregado.

Comportamiento Aceptable:

- ◆ El sistema registra en la base de datos, el cambio de estado del vehículo de entregado a disponible.
- ◆ El sistema muestra un mensaje de felicitaciones por la puntualidad en la entrega.

Nombre del Grupo de prueba: Ingreso de Vehículo nuevo

Numero de prueba: 2.3

Pre-requisitos:

El administrador de la compañía conoce su password y tiene una foto del vehículo a ingresar.

Instrucciones de Configuración:

- ◆ El vehículo se encuentra en excelentes condiciones para el alquiler.
- ◆ El administrador considera tener más vehículos a disposición en la Oficina.

Instrucciones de Prueba:

- ◆ El administrador escoge la opción Ingresar nuevo vehículo

Comportamiento Aceptable:

- ◆ El sistema tiene otro vehículo disponible para alquilar.

Nombre del Grupo de prueba: Baja de Vehículo usado

Numero de prueba: 2.4

Pre-requisitos:

El administrador considera que un vehiculo debe ser dado de baja porque se encuentra en malas condiciones para el alquiler.

El administrador conoce su password.

Instrucciones de Prueba:

- ◆ El administrador ingresa a la opción Administración del Sistema y coloca su password.
- ◆ Ingresa a la opcion Mantenimiento de vehiculos, luego a la opcion Dar de baja a vehiculos.

Comportamiento aceptable:

- ◆ El vehiculo no se encuentra disponible para el alquiler.

Conclusiones

1. Considerando al lenguaje Java como una herramienta poderosa, pero de reciente aplicación en nuestro medio, el aprendizaje de la misma es un factor a considerar.
2. Los detalles de protocolos de comunicación son transparentes para el programador, gracias a la utilización de **ORB** ofrecido por Corba.
3. Se facilita la realización de transacciones y procesos, ya que permite correr aplicaciones que por su complejidad no se podían realizar en Internet. Ahora no nos limitamos al ingreso de simples formas cuyos datos se harán persistentes en una base de datos.
4. Constituye una excelente fuente de aprendizaje, ya que el tipo de tecnología y arquitectura usada es nueva en nuestro medio y esta avanzando a grandes pasos, es decir, se amplían enormemente las posibilidades de aplicación de la arquitectura y por ende enormes fuentes de trabajo.
5. De acuerdo a la tendencia actual, la arquitectura usada en la realización de esta tesis se constituirá en algo tan común como la invocación de algún método o función local, en lo que a programación se refiere.

Glosario de Términos

DCOM.- Modelo de Objetos componentes distribuidos. Es la arquitectura e implementación de tecnología de Microsoft.

CORBA.- Common Object Request Broker Architecture. Es la especificación de OMG para la arquitectura de ORBs, no la implementación.

DCE.- Distributed Computing Environment. Es la arquitectura de computación distribuida desarrollada por la OSF antes de CORBA.

DII.- Dynamic Invocation Interface. Un API del lado del cliente, usado para generar mensajes a través de la red, es independiente de cualquier IDL, aunque es más tedioso de usar que cualquier IDL.

DSI.- Dynamic Skeleton Interface. Un API del lado del servidor, usado para recibir mensajes de la red, es independiente de cualquier IDL, aunque es más tedioso de usar que cualquier IDL.

ESIOP.- Environment Specific Inter-ORB Protocol, La implementación de GIOP para ambientes que no son TCP/IP, e.j. DCE.

GIOP.- General Inter-ORB Protocol. La especificación de alto nivel de CORBA 2.0, permite a los clientes y servidores usar diferentes ORBs para interoperar.

IDL.- Interface Definition Language. Es el lenguaje independiente del lenguaje de programación utilizado para escribir las interfaces CORBA.

IFR.- Interface Repository. Es el servicio CORBA que almacena meta-data acerca de las interfaces IDL.

IIOB.- Internet Inter-ORB Protocol. Es la implementación de GIOP para TCP/IP.

IOR.- Interoperable Object Reference. Una referencia de objeto (OR) es la forma en la que se identifica a un objeto CORBA. Es el formato estándar para representar una referencia de objeto para todos los proveedores de ORB.

OMG.- Object Management Group. Es una organización sin fines de lucro que dio vida a CORBA.

OR.- Object Reference. Es la forma en que se identifican a los objetos CORBA.

ORB.- Object Request Broker. Permite la interoperabilidad y la comunicación de los objetos clientes y servidores.

OSF.- Open Software Foundation. Creador de DCE.

SII.- Static Invocation Interface. Es el API del lado del cliente que sirve para generar mensajes en la red que se basan en los stubs generados por el compilador IDL para una interface dada.

SSI.- Static Skeleton Interface. Es el API del lado del servidor que recibe los mensajes.

TCP/IP.- Transport Control Protocol/Internet Protocol. Un protocolo de red de bajo nivel usado para Internet y muchas otras redes.

Bibliografía

1. I. CABRERA, “Tecnología Cliente Servidor con Arquitectura CORBA” (Tesis, Facultad de Ingeniería Eléctrica y Computación, Escuela Superior Politécnica del Litoral, 1998).
2. ROBERT ORFALI & DAVID HARKEY, “Client/Server Programming with Java and CORBA” (2da. Edición; New York: John Wiley & Sons Inc, 1998), Capítulos 4 – 5 – 19 – 20 – 21 – 35 – 36 – 37 – 38.
3. J. PICON, C. EDWARDS & G. SCENINI, “Using VisualAge for Java Enterprise Version 2 to Develop CORBA and EJB Applications” (1era. Edición; New York: Red book de IBM, <http://www.redbooks.ibm.com>, 1998), Capítulos 6 – 7 – 8 – 9 – 10.
4. O. GRAF, A. KOTZEN, O. TAKAGIWA & U. WAHLI, “VisualAge for Java Enterprise Version 2: Data Access Beans – Servlets – CICS Connector” (1era. Edición; New York: Red book de IBM, <http://www.redbooks.ibm.com>, 1998), Capítulo 3.

5. IBM, "Servlet Builder" (1era. Edición; New York: Red book de IBM, 1998).
6. R. HARKEY D. & EDWARDS J., Essential Client/Server Survival Guide (2da. Edición; New York: John Wiley & Sons, Inc, 1996).
7. R. HARKEY D. & EDWARDS J., Instant Corba (1era. Edición; New York: John Wiley & Sons, Inc, 1997).
8. Documentación asociada con cada uno de los productos utilizados en la realización del presente proyecto.